DESIGN AND IMPLEMENTATION OF A MULTI-AGENT RAG SYSTEM FOR COLLEGE-LEVEL KNOWLEDGE PROCESSING

Chelluri Saiteja[1], Chollangi Meghana Harini[2], Chukka Koteswara Rao[3], Galla Venkataswamy[4]
[123]B.Tech Final Year Students, [4]Assistant Professor
[1234]Department of Computer Science and Engineering (Data Science)
Raghu Institute of Technology , Visakhapatnam

Abstract

This paper presents the design and implementation of a Multi-Agent Retrieval-Augmented Generation (RAG) system for college-level academic question answering. Students often face challenges in retrieving accurate and context-specific information from large academic textbooks due to limitations in traditional keyword-based search methods. To address this issue, the proposed system integrates a multi-agent architecture consisting of a Query Understanding Agent, Retrieval Agent, Answer Generation Agent, and Verification Agent. The system processes academic PDF documents through a structured pipeline involving text extraction, preprocessing, chunking, embedding generation, and vector indexing using a Qdrant database. Semantic retrieval is performed using BAAI/bge-large-en-v1.5 embeddings, while answer generation is handled by the Gemma 2 9B language model, ensuring responses are grounded in retrieved context. A Streamlit-based web interface allows users to interact with the system through natural language queries, providing answers along with confidence scores and source citations. Experimental results demonstrate 90% subject detection accuracy, 84% retrieval precision, and 80% answer accuracy, with average response time below 30 seconds. The proposed system provides a scalable, reliable, and explainable solution for academic knowledge processing.

Keywords: Multi-Agent Systems, Retrieval-Augmented Generation (RAG), Semantic Search, Vector Database, Large Language Models, Educational AI

1. Introduction

Engineering students frequently work with large volumes of academic content across subjects such as Database Management Systems, Machine Learning, and Operating Systems. Extracting relevant information from these resources is often time-consuming and inefficient. Traditional search systems rely on keyword matching, which fails to capture semantic meaning. As a result, students struggle to find accurate answers when queries are phrased differently from textbook content. On the other hand, general-purpose AI tools can generate answers but often lack reliability due to hallucination and absence of source grounding. Retrieval-Augmented Generation (RAG) has emerged as an effective approach to overcome these issues by combining document retrieval with language generation. However, basic RAG systems lack advanced features such as query understanding, subject filtering, and answer verification. To address these limitations, this paper proposes a Multi-Agent RAG system that introduces specialized agents for each stage of processing. The system ensures accurate retrieval, context-based answer generation, and reliability through confidence scoring. The main objective of this work is to develop a system that improves learning efficiency by providing precise, explainable, and trustworthy academic answers.

2. Review of Literature

Retrieval-Augmented Generation was introduced by Lewis et al. (2020) to improve the factual accuracy of language models by integrating external document retrieval. This approach reduces hallucination and enhances reliability. Dense Passage Retrieval, proposed by Karpukhin et al. (2020), improved semantic search by representing text as dense vectors, enabling retrieval based on meaning rather than keywords.

Vector databases such as Qdrant utilize efficient algorithms like HNSW (Malkov & Yashunin, 2018) for high-speed similarity search. These systems are widely used in modern AI applications for scalable retrieval. Multi-agent systems, as described by Wooldridge (2009), enable complex tasks to be divided into

smaller components handled by specialized agents. This improves modularity, scalability, and system performance.

Recent embedding models like BAAI/bge-large-en-v1.5 have shown strong performance in semantic understanding, while instruction-tuned language models like Gemma improve controlled answer generation. Despite these advancements, existing systems lack integrated query understanding, domain filtering, and verification mechanisms. The proposed system addresses these gaps using a structured multi-agent approach.

3. Methodology

We followed a structured development process to design and implement the Multi-Agent Retrieval-Augmented Generation (RAG) system for academic question answering. The methodology integrates document processing, semantic retrieval, multi-agent coordination, and response verification to ensure accurate and reliable results.

3.1 Data Collection and Preprocessing : The system uses academic PDF documents from subjects such as Database Management Systems, Machine Learning, and Operating Systems. These documents are processed using PyMuPDF for text extraction.the extracted text is cleaned by removing unwanted elements such as headers, footers, page numbers, and special characters. The cleaned text is then divided into smaller chunks of approximately 500 words with an overlap of 100 words. This overlapping strategy preserves contextual continuity and improves retrieval accuracy. Each chunk is stored along with metadata such as subject, document name, and page number for efficient access during retrieval.

3.2 Embedding and Vector Storage: Each text chunk is converted into a numerical vector representation using the BAAI/bge-large-en-v1.5 embedding model. These embeddings capture the semantic meaning of the text and enable similarity-based search**.** The generated embeddings are stored in a Qdrant vector database. The database uses cosine similarity for efficient retrieval of relevant content. Subject-based filtering is also applied to improve the relevance of search results.

3.3 System Architecture : The overall system architecture is shown in image (i) , which illustrates the complete workflow of the Multi-Agent RAG system.

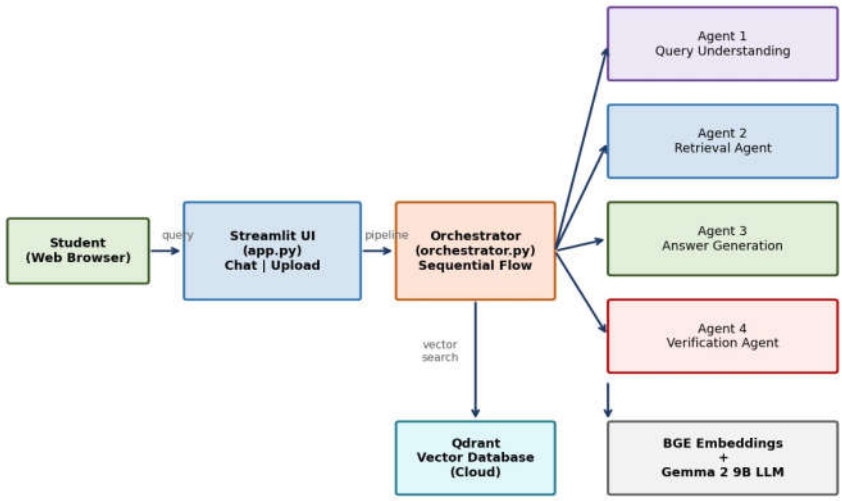

Image (i) : System Architecture of the Multi-Agent RAG Study Assistant

The system consists of four main components:

- User Interface: A Streamlit-based interface where users can submit queries and upload documents
- Orchestrator: Manages the flow of execution between different agents
- Agent Layer: Includes four specialized agents for processing queries

- Storage and Model Layer: Consists of the Qdrant vector database, embedding model, and language model

This layered architecture ensures modularity, scalability, and efficient processing of user queries.

3.4 Multi-Agent Processing Pipeline

The system uses a multi-agent pipeline to process user queries in a sequential manner:

- Query Understanding Agent:
  Identifies the subject of the query and refines it for better retrieval
- Retrieval Agent:
  Converts the query into an embedding and retrieves relevant text chunks from the vector database
- Answer Generation Agent:
  Generates answers using the language model based only on the retrieved context
- Verification Agent:
  Evaluates the generated answer and assigns a confidence score

Each agent performs a specific function, ensuring a modular and efficient system design.

3.5 Verification Mechanism: The verification mechanism ensures the reliability of generated responses. It evaluates the answer using two key metrics:

- Retrieval similarity score
- Lexical overlap between the generated answer and retrieved content

Based on these metrics, the system assigns confidence levels such as HIGH, MEDIUM, LOW, and NOT_FOUND. This helps reduce incorrect or misleading responses and improves user trust.

3.6 Web Integration and Deployment: A web application is developed using Streamlit to provide an interactive and user-friendly interface. The system allows users to:

- Enter natural language queries
- View generated answers with confidence scores
- Access source citations
- Upload new PDF documents

The system is deployed on standard hardware with optional GPU support for faster processing. It supports real-time interaction and dynamic expansion of the knowledge base.

4. Results

The proposed Multi-Agent RAG system was evaluated using academic queries from subjects such as Database Management Systems, Machine Learning, and Operating Systems. The system was tested for accuracy, response quality, and usability.

4.1 System Output Interface: The system provides an interactive web-based interface for answering academic queries. A sample output generated by the system is shown in images (ii), (iii)

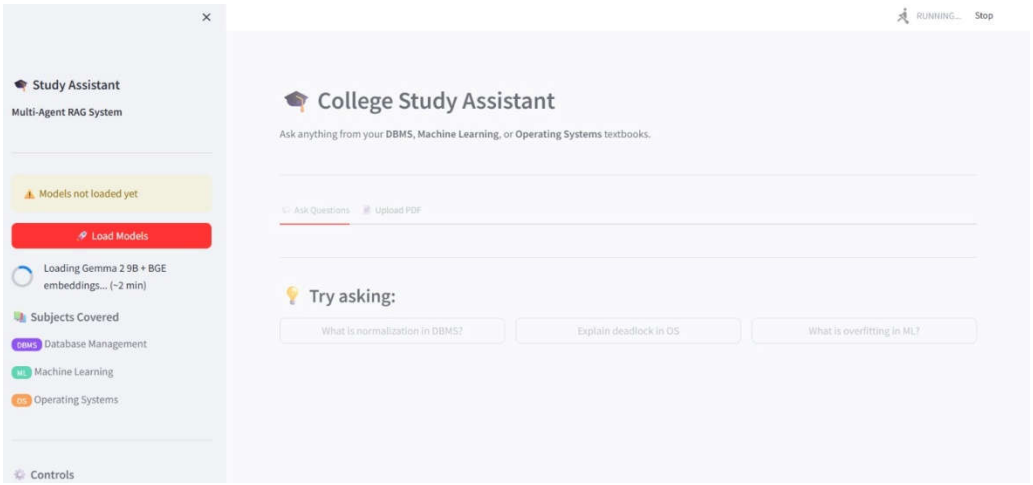

Image : (ii)

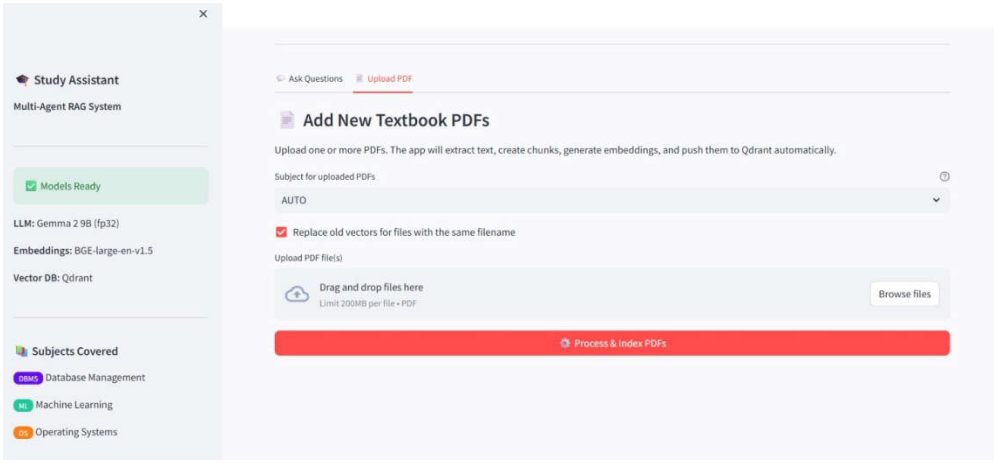

Image : (iii)

Images (ii) & (iii): Output Interface Showing Generated Answer, Confidence Score, and Source Citations

The interface displays:

- The generated answer based on retrieved content
- A confidence score indicating the reliability of the response
- Subject classification of the query
- Source citations including document name and page references

This ensures that users can verify the correctness of the response and understand the source of information.

4.2 Performance Evaluation: The system was evaluated using a set of academic queries across multiple subjects. The performance metrics are summarized below:

Metric	Value
Subject Detection Accuracy	90%
Retrieval Precision	84%
Answer Accuracy	80%
Confidence Accuracy	87%
Average Response Time	~18 seconds

These results indicate that the system performs effectively in retrieving relevant information and generating accurate answers.

4.3 Output Analysis

The system produces context-based and source-grounded answers for academic queries. For well-defined queries, the system generates responses with high confidence and accurate explanations. For queries where relevant information is not available in the dataset, the system assigns a **NOT_FOUND** confidence level, ensuring that no incorrect or misleading answers are generated.

4.4 System Reliability

The system was tested under different scenarios, including:

- Subject-specific queries
- Conceptual and descriptive questions
- Out-of-domain queries

The results show that:

- Relevant queries produce accurate and reliable answers
- The multi-agent pipeline ensures proper query understanding and retrieval
- The verification mechanism improves trust by assigning appropriate confidence scores

5. Discussion

The results demonstrate that the Multi-Agent RAG system significantly improves over traditional approaches. The use of semantic embeddings allows better retrieval compared to keyword-based systems. The multi-agent architecture enhances modularity and ensures each stage is optimized independently. The verification mechanism plays a crucial role in improving reliability by assigning confidence scores.

Compared to basic RAG systems, the proposed model provides:

- Better query understanding
- More relevant retrieval
- Controlled answer generation
- Reliable evaluation of responses

However, the system depends on the quality and coverage of the dataset. Performance may decrease if relevant content is not present in the knowledge base.

6. Conclusion

This paper presented a Multi-Agent RAG system for academic knowledge processing. The system integrates semantic retrieval, language generation, and verification to provide accurate and reliable answers. The multi-agent architecture improves performance by dividing tasks into specialized components. The system demonstrates strong results in terms of accuracy, efficiency, and usability. This work highlights the potential of combining RAG with multi-agent systems for real-world educational applications.

7. References

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W. T., Rocktäschel, T., Riedel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems, 33*, 9459–9474. https://doi.org/10.48550/arXiv.2005.11401

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. T. (2020). Dense passage retrieval for open-domain question answering. *Proceedings of EMNLP*. https://doi.org/10.18653/v1/2020.emnlp-main.550

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42*(4), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems, 35*. https://doi.org/10.48550/arXiv.2201.11903

Yao, S., Zhao, J., Yu, Y., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. *Proceedings of ICLR*. https://doi.org/10.48550/arXiv.2210.03629

Manakul, P., Liusie, J., & Gales, M. J. (2023). SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. *Proceedings of EMNLP*. https://doi.org/10.48550/arXiv.2303.08896

Xiao, S., Liu, Z., Zhang, P., & Muennighoff, N. (2023). C-Pack: Packaged resources to advance general Chinese embedding. *arXiv preprint*. https://doi.org/10.48550/arXiv.2309.07597

Google DeepMind. (2024). Gemma: Open models based on Gemini research and technology. *arXiv preprint*. https://doi.org/10.48550/arXiv.2403.08295

Wooldridge, M. (2009). *An introduction to multiagent systems* (2nd ed.). John Wiley & Sons. https://doi.org/10.1002/9780470519462

Qdrant Team. (2024). Qdrant vector database documentation. https://qdrant.tech/documentation/

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *Proceedings of EMNLP*. https://doi.org/10.18653/v1/D19-1410

James, E. S., Espinosa-Anke, L., & Schockaert, L. (2023). RAGAS: Automated evaluation of retrieval augmented generation. *arXiv preprint*. https://doi.org/10.48550/arXiv.2309.15217

Hugging Face Team. (2024). Transformers: State-of-the-art machine learning for PyTorch, TensorFlow, and JAX. https://huggingface.co/docs/transformers

Streamlit Inc. (2024). Streamlit documentation. https://docs.streamlit.io

PyMuPDF Team. (2024). PyMuPDF documentation. https://pymupdf.readthedocs.io

Guo, Z., Schlöter, Z., Schlichtkrull, M., & Vlachos, A. (2022). A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics, 10*, 178–206. https://doi.org/10.1162/tacl_a_00454

Johnson, J., Douze, M., & Jégou, H. (2021). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data, 7*(3), 535–547. https://doi.org/10.1109/TBDATA.2019.2921572