International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

AI POWERED MULTIMODAL ASSISTANT: A Graph-Based Multi-Agent Framework Integrating Retrieval-Augmented Generation for Intelligent Document Processing and Content Generation

Pusarla Bhanu Teja, Information Technology, MVGR College of Engineering,

Latchubhukta Srimani, Information Technology, MVGR College of Engineering,

Pammi Venkata Reddy, Information Technology, MVGR College of Engineering,

Korada Aravind, Information Technology, MVGR College of Engineering.

Ms.D.Sheetal Sharma, Assistant Professor, Information Technology, MVGR College of Engineering

---

Abstract - The paper presents the design and development of an AI Powered Multimodal Assistant that integrates multiple intelligent functionalities into a single platform. The system is developed using modern technologies such as LangGraph, LangChain, Retrieval-Augmented Generation (RAG), and Large Language Models (LLMs). It follows a modular and graph-based workflow where tasks are divided into smaller steps for efficient processing. The system supports multiple use cases such as chatbot interaction, PDF-based question answering, web search retrieves relevant data before generating responses, which improves accuracy and reduces incorrect outputs. The Streamlit-based interface provides a simple and interactive user experience. The proposed system improves efficiency, scalability, and usability compared to traditional AI systems.

Keywords: Artificial Intelligence, LangGraph, RAG, LLM, Chatbot, Multimodal System

1. Introduction

Artificial Intelligence (AI) has developed through its recent progress to create intelligent systems that function in actual real-world environments. The conventional AI systems show two main problems because they cannot execute multiple tasks at once and they fail to connect with outside data sources. The research presents an AI Powered Multimodel Assistant because it combines different features into one platform. The system uses LangGraph and LangChain and Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs) technologies to transform user input into contextually relevant output. The system uses a graph-based workflow architecture which enables structured task execution through efficient processing methods. The system supports applications such as chatbot interaction, document-based question answering, web search, and blog generation. The system uses RAG to enhance response precision while decreasing the frequency of incorrect results. The proposed system aims to provide a scalable, efficient, and user-friendly solution for complex AI-driven tasks.

2. Related Work

More recently research in conversational AI and information retrieval have moved away from solely keyword-based methods and instead integrate semantic understanding with context. Transformer models like BERT are often the default baseline for language based problems and are able to provide context aware embeddings, hence being very useful for information retrieval tasks such as document search and question answering where context is key [2].

RAG expands on this concept by combining semantic search and generative models. The work of Patrick Lewis et al shows that integrating outside knowledge sources within language models lead to improvements in knowledge based tasks [1]. The concept fits document-based systems as relevant documents need to be retrieved first to generate an appropriate answer. Vector similarity search with methods such as FAISS are suitable for this as they can search quickly and in a scalable way across large numbers of high dimensional embeddings [3].

Large language models trained by OpenAI provide strong capabilities in text generation, summarization and reasoning which can be applied to chatbot systems and automatic content generation [7]. These models can create very human like output and improve user interaction in AI systems greatly. Combining document context with layout can significantly help improve information extraction and are used in documents with

structured layout like PDFs and forms using models such as LayoutLM [6].

Finally LangChain and LangGraph provides frameworks to orchestrate complex multi agent systems that can run in a modular and scalable way by defining states in a graph based approach [4][5].

3. Proposed System

Proposed system is an Agentic AI platform that utilizes LangGraph to tackle the limitations of traditional AI with structured workflows and intelligent decision-making. The platform incorporates technologies like LangGraph, LangChain, RAG (Retrieval Augmented Generation), and large language models to provide an efficient and scalable solution. It is built around a graph structure, with tasks representing the nodes and the edges connecting the nodes of the graph allow the controlled execution of tasks, modular design, and multi-step reasoning.

The response accuracy is improved by using RAG: the user input can be converted to embeddings and stored in a vector database using FAISS. The information relevant to the user query can be retrieved based on semantic similarity and presented along with the generation capabilities of the LLMs to provide context-specific response and avoid hallucination. Streamlit is used to develop a unified web interface that can be used for document-based question answering, blog generation and conversational chat while retaining the context across the session. The whole system is modular, scalable and efficient and can be used for intelligent document processing, content generation and knowledge retrieval.

4. Methodology

The system is designed with a systematic and structured workflow for intelligent document processing, information retrieval, and content generation. The user inputs information into the system through the web interface (text, PDF documents, URLs or multimedia sources like web content and transcripts) and then into the system for preprocessing and further processing like text extraction, text cleaning and normalization. To effectively understand the context and semantic relationships between the text, the system utilizes embeddings from Hugging Face pre-trained models that represent the textual data into high dimensional vectors. The vectors are stored in a vector store using FAISS to retrieve information through similarity matching. The system retrieves the required information from the vector store and passes it to large language models provided by OpenAI to generate a response with accuracy and reliability, minimizing hallucination by utilizing Retrieval Augmented Generation.

4.1 Data Processing and Preprocessing

The entire workflow is initiated with the ingesting of input from multiple sources such as text, PDF documents, URLs or structures from web content. This ingested content is then pre-processed, including tokenization, lower-casing, stop-word removal, lemmatization and noise removal in order to eliminate anomalies and irregularities and present a consistent textual data.

4.2 Feature Representation (Embeddings)

Utilized from the existing Hugging Face models, embeddings are used to convert text to dense vector representations so as to capture the sematic relationships between words and sentences and present a coherent representation of the meaning of the input data which also applies to user query and document data so that a comparison between user query and the documents could be easily performed and efficient result could be obtained.

4.3 Retrieval-Augmented Generation (RAG)

To achieve the goal of accurate and reliable answer generated, the RAG approach is used to retrieve relevant pieces of information and feed it into large language models. These pieces of information are first stored in a vector store using FAISS for efficient similarity search. Once the user gives a query, relevant pieces are fetched and then combined to generate an accurate and contextually appropriate answer, which minimizes the chance of hallucination.

4.4 Graph-Based Multi-Agent Workflow

The system is built around a graph-based workflow built with LangGraph. In this architecture, the different tasks are mapped to the nodes and edges in a directed graph. The overall work is performed by different agents specialized for the task (input processing, retrieving documents, generating an answer etc.). These agents communicate via a set of signals defined in the graph. The LangChain framework is used to build interfaces between these modules.

4.5 Functional Modules

The designed system provides a suite of functional modules to cover a wide range of user requirements which are document based Q&A, conversational chatbot, text summarization, blog/content generation, web content analysis, transcript based Q&A etc.

4.6 User Interface

The user interface is built in Streamlit and provides real time interactivity to the users to work with the system. The overall flow, process flow diagram and block diagram present a comprehensive understanding on how the data moves in the system.

4.7 System Performance and Optimization

The system benefits from FAISS's efficiently scalability retrieval process, Hugging Face's high accuracy embeddings and graph-based multi-agent workflow that contributes to the low processing latency, system scalability and the output quality using various methods of structured validation.

5. System Design

The System Design displays a view of the platform we plan to build. We show the process flow beginning with a user's input, progressing through preprocessing, then using Hugging Face to generate embeddings andFAISSto retrieve it. Finally, the model combines these components by connecting large language models via Groq, for a highly accurate, contextually aware response, using a RAG system.

5.1 Block diagram

Figure 1 displays a block diagram which shows the complete system architecture design. Users start the process by providing input which can include text and PDF documents and web URLs. The system preprocesses input data to extract valuable information which it transforms into embedding vectors. The system stores these embedding vectors in a vector database to support fast similarity-based information retrieval. The system uses RAG to fetch relevant data when users ask for information and delivers it to the language model which creates answers.

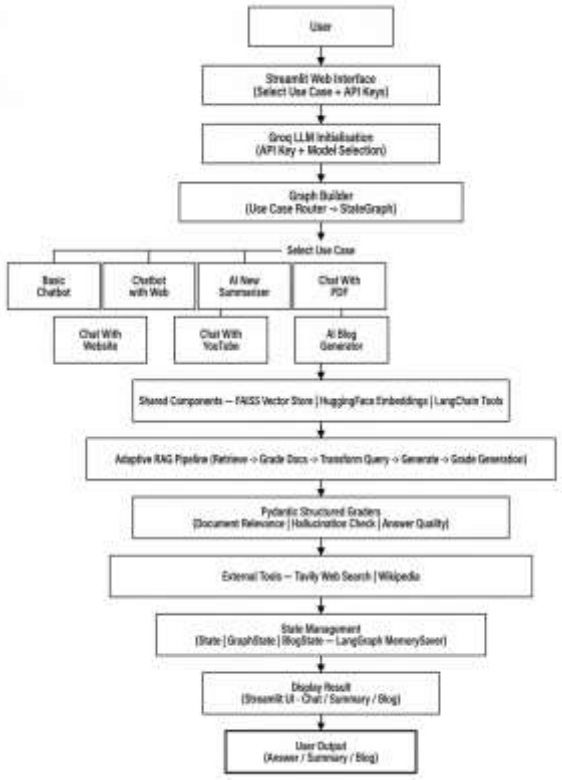

Fig -1: Block diagram

5.2 Workflow

The system diagram in Fig. 2 uses a graph-based method to show how the system operates. The process begins with the system retrieving pertinent information needed to answer the user question. The system examines retrieved documents to establish their value. The system changes the query and starts the retrieval process again when the documents prove to be unhelpful. The system uses LLMs to generate answers when it discovers useful information. The system achieves precise results through an iterative method which produces trustworthy results.

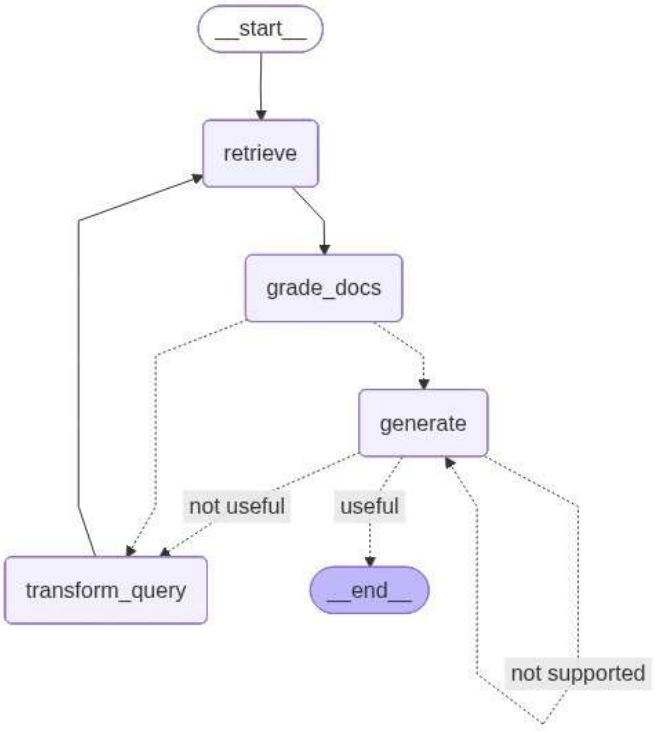

Fig -2: Workflow Process

6. Testing Results

6.1 Chatbot Testing Results

Fig 6.1 Chatbot Testing Results

After testing the Basic Chatbot and Chatbot with Web Search modules, the following results were obtained. The Basic Chatbot was tested with general queries to verify its response accuracy and memory retention across multiple turns. The Chatbot with Web Search was tested with real-time queries to verify its ability to fetch live information using Tavily Search and Wikipedia tools. The system successfully maintained conversation history across multiple interactions using LangChain's message history mechanism. The chatbot correctly identified when to use tools and when to answer directly from the language model.

6.2 Chat With PDF / Website Output

Fig 6.2 Chat With PDF / Website Output

The RAG-based modules — Chat With PDF and Chat With Website — were tested by providing respective inputs and asking relevant questions. For Chat With PDF, a PDF document was uploaded through the Streamlit interface and questions were asked based on the document content. The system successfully loaded the PDF, split it into chunks, created embeddings using HuggingFace, stored them in a FAISS vector store, and retrieved the most relevant chunks to generate accurate answers. For Chat With Website, one or more website URLs were entered and the system scraped the content, processed it, and answered questions based on the website data. In both cases, the Adaptive RAG pipeline correctly graded retrieved documents for relevance, rewrote queries when necessary, and verified the generated answer for hallucinations before displaying it to the user.

6.3 AI Blog Generator Output

Fig 6.3.1 AI Blog Generator Output

Fig 6.3.2 AI Blog Generator Output

The AI Blog and clicking the Generate button. The system invoked the LangGraph blog generation graph which first generated a creative, SEO-friendly blog title using the Groq LLM and then generated detailed long-form blog content in

Markdown format covering the topic comprehensively. The topic-only flow was tested and the output contained a proper title, introduction, detailed sections, and conclusion. Additionally, the language translation flow was tested by selecting Hindi and French as target languages. The system successfully routed the generated blog content through the translation node and produced accurate translated output while maintaining the original formatting and tone. Both flows — topic-only and language-based — worked correctly without any errors.

7. Discussion

This experimental evaluation showed the potential to fall victim to hallucination, reduced factual correctness, and other undesirable properties that can result from the application of large language models in the real world. The implemented solution's use of a Retrieval-Augmented Generation (RAG) model provides reliability by retrieving relevant information from which the LLM can formulate an answer, grounded on facts from an external source. Using Hugging Face embeddings to ensure accuracy, and efficient FAISS similarity search to do so quickly allows for accurate, context-based retrieval. An additional design element was to implement a graph-based multi-agent architecture using LangGraph. This allows the separation of complicated tasks into a more manageable format, and results in improved modularity, scalability and maintainability. Multiple applications, including conversations between a chatbot and a user, questions and answers derived from a given document and blog creation, are successfully integrated. This model also leverages large language models through Groq to improve performance and reduce response times. Overall this implementation represents a highly effective, scalable model for real-world application, which could be further improved with increased data and full user analysis.

8. Conclusion

In this paper, an intelligent AI-based multi-agent platform was developed for the purpose of processing documents, generating content, and querying for information. The system incorporates state-of-the-art tools such as LangGraph, LangChain and RAG for accurate and context-relevant results. Embeddings from Hugging Face along with fast similarity search through FAISS facilitate accurate information retrieval and reduce hallucination. A graph-based multi-agent framework was utilized for the ease of modularity, scalability and complex workflows management. A Groq LLM was also employed to increase response speed and improve the overall efficiency of the system.

The platform supports various real-world applications including conversation chat, document-based question answering, blog generation etc., all integrated under a single framework. Experiments results show that the fusion of retrieval and generation approaches proves reliable and semantically accurate. All in all, the developed system is scalable and user friendly for any modern AI application, further improvements include enhanced performance through more data, specific fine-tuning for different domains and real-time evaluation by users.

Publication Note

This manuscript presents the AI-Powered Multimodal Assistant

In a concise, system-focused format. Prior to submission ensure alignment with the target journal's template and verify authorship and ethical compliance.

References

[1] Lewis, P., Perez, E., Piktus, A., Karpukhin, V., Goyal, N., etc. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in Neural Information Processing Systems, 33, 9459–9474. Https://papers.nips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[2] Devlin, J., Chang, M.W., Lee, K., Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of NAACL-HLT 2019. Https://doi.org/10.18653/v1/N19-1423

[3] Johnson, J., Douze, M., Jgou, H. (2017). FAISS: A library for efficient similarity search and clustering of dense vectors. ArXiv preprint arXiv:1702.08734. Https://arxiv.org/abs/1702.08734

[4] LangChain. (2023). LangChain: Building applications with large language models. Retrieved on April 2nd, 2026 from https://www.langchain.com

[5] LangGraph. (2024). LangGraph: Graph-based workflow orchestration for AI systems. Retrieved on April 2nd, 2026 from https://www.langgraph.dev

Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., Zhou, M. (2020). [6] LayoutLM: Pre-training of text and layout for document image understanding. In Proceedings of the 28th ACM International Conference on Multimedia, 28, 2483–2491. Https://doi.org/10.1145/3394486.3403172

[7]OpenAI. (2023). GPT-4 technical report. ArXiv preprint arXiv:2303.08774. Https://arxiv.org/abs/2303.08774