International Journal of Scientific Research & Engineering Trends
Volume 10, Issue 5, Sept-Oct-2024, ISSN (Online): 2395-566X

Advanced Multi Model RAG Application

Professor Disha Nagpure, Sujal Pore, Shardul Deshmukh, Aditya Suryawanshi
Department of Artificial Intelligence and Machine Learning,
Alard College of Engineering and Management, Marunji, Pune, Maharashtra, India

Abstract- This paper presents a modular, context-aware multimodal Retrieval-Augmented Generation (RAG) application that leverages both chain-based and agentic execution strategies. Powered by Gemini 1.5 Flash as the core language model, the system integrates Langchain and Langsmith frameworks to enable dynamic document retrieval, task orchestration, and seamless handling of multiple data sources. Key features include a YouTube summarizer using transcript APIs, real-time web search via the Tavily search tool, and support for text, image, and audio inputs, with OpenAI's Whisper model for speech-to-text conversion. The application's contextual awareness is enhanced by chat memory fallback functions, ensuring continuous, coherent interaction across sessions. Additionally, vector databases are employed for efficient multimodal retrieval. This system represents a significant advancement in RAG applications, offering flexibility, scalability, and adaptability across various input modalities and real-time tasks.

Index Terms- Multimodal Retrieval-Augmented Generation (RAG), Context-aware AI, Gemini 1.5 Flash, Langchain, Langsmith, Chain-based execution, Agentic execution, Modular architecture, YouTube summarizer, Transcript API, Tavily search tool, Web search, Image-based chat, Audio input, OpenAI Whisper model, Chat memory fallback, Vector databases, Speech-to-text, Real-time data retrieval, Document loaders, AI agents.

I. INTRODUCTION

The rise of advanced artificial intelligence technologies has paved the way for the development of sophisticated multimodal applications, transforming the landscape of AI-powered systems. One area experiencing significant innovation is Retrieval-Augmented Generation (RAG), where the integration of multimodal inputs, contextual awareness, and realtime capabilities offers considerable potential for various use cases. Traditional RAG applications often face limitations in handling diverse data sources and maintaining context across interactions. To address these challenges, this research explores the implementation of an advanced multimodal RAG application using Gemini 1.5 Flash as the core language model, incorporating both chainbased and agentic execution approaches.

This application introduces several key features designed to enhance the flexibility and functionality of RAG systems. By utilizing Google Generative AI embeddings for embedding text-based documents and OpenCLIP embeddings for image-based content, the system enables seamless multimodal interaction. Additionally, the use of Langchain and Langsmith frameworks facilitates dynamic task orchestration and document retrieval, while the integration of the Tavily search tool provides real-time web search functionality. A YouTube summarizer using transcript APIs further expands the application's utility by enabling efficient extraction of video content.

One of the unique aspects of this system is its ability to handle audio inputs through OpenAI's Whisper model, which converts speech to text, and its contextual awareness supported by chat memory and fallback functions, ensuring coherent interactions across sessions. To enhance retrieval efficiency, the application also leverages vector databases, making it a robust tool capable of managing complex data sources in real time. This research delves into the design, architecture, and key functionalities of this modular RAG system, demonstrating how it advances the state of AIdriven multimodal applications.

1. Traditional Chat Application

Traditional chat applications, while effective for basic conversational tasks, face several limitations in handling complex, multimodal inputs and maintaining contextual continuity across interactions. These systems typically rely on simple text-based exchanges and lack advanced features such as real-time data retrieval and the ability to process diverse input formats like images or audio. Traditional chat applications may face the following problems:

Limited Multimodal Support: Most traditional chat applications are text-based, making it difficult to integrate image, audio, or video data into the conversation flow.

Context Management: Maintaining context over extended conversations remains a challenge. Without advanced memory fallback functions, traditional chat systems often lose track of previous exchanges, leading to disjointed user experiences.

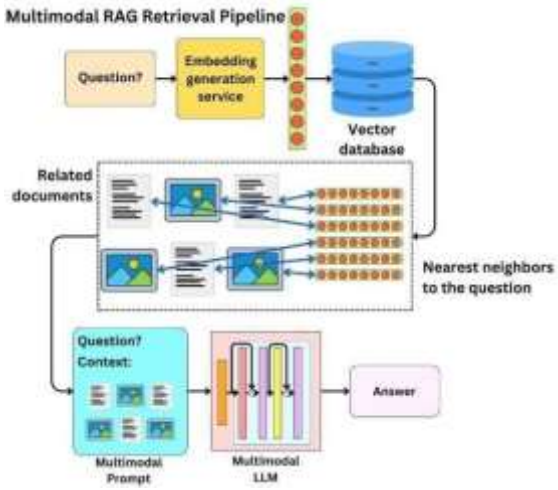
Real-time Data Retrieval: Traditional chat applications generally lack real-time web search or integration with external data sources, limiting their ability to provide timely, accurate information in response to queries.

Static Architecture: The majority of traditional chat applications rely on a static, monolithic architecture, making it difficult to scale or adapt to more complex user requirements, such as dynamic task management or real-time updates.

Embeddings Limitations: In traditional chat applications, the embedding mechanisms for processing input are often rudimentary, typically confined to handling only text data. This limitation hinders their ability to fully leverage the richness of multimodal inputs like images or voice.

Basic Interaction Models: Traditional chat applications generally follow a simple request-response model, with no capacity for agent-based task management or chain execution, resulting in limited functionality when it comes to handling complex workflows.

2. What is a Multi Model RAG?



Multimodal Retrieval-Augmented Generation (RAG) is an advanced framework that integrates various data modalities—such as text, images, and audio—to enhance the capabilities of AI systems. This approach combines traditional text-based generation with retrieval mechanisms that allow the system to access and incorporate relevant external information dynamically. By utilizing models like Gemini 1.5 Flash, multimodal RAG applications can process and respond to diverse inputs, providing richer and more contextually relevant interactions. The integration of embeddings, such as Google Generative AI for text and OpenCLIP for images, facilitates seamless interaction across different content types. Additionally, the architecture often incorporates features like real-time web search, chat memory, callback functions, and agent-based task management, enabling a cohesive user experience that adapts to complex queries and workflows.

This innovative approach significantly enhances the versatility and effectiveness of AI-driven applications, making them suitable for a wide range of use cases.

3. Working of the Multimodal RAG Application

Multimodal RAG Application Overview: The Multimodal RAG application combines various input modalities, including text, images, and audio, to provide comprehensive answers using advanced language models and tools. The application leverages document retrieval systems, vector databases, and generative AI to create an intelligent chatbot interface.

User Interaction

Uploading Files: Users can upload various document types (PDFs, images, CSVs) through the interface. The application processes these files and extracts information for later retrieval.

Chat Interface: Users can interact with the chatbot by typing queries in a user-friendly chat interface. The application maintains a session state to keep track of the conversation history.

Processing Inputs

File Processing: Upon uploading files, the application utilizes loaders (e.g., PDFMinerLoader for PDFs, CSVLoader for CSVs) to convert documents into a format suitable for embedding and searching. Images are stored in a Chroma database for visual queries. Generating Embeddings: The application generates embeddings for both textual and image data using Google Generative AI embeddings and OpenCLIP embedding functions. These embeddings facilitate efficient retrieval of relevant information based on user queries.

Query Handling

Text Queries: When a user submits a query, the application evaluates the input and utilizes structured agents to determine the best tools for generating a response. It can call external APIs (like Tavily) or search the vector store.

Image Queries: If the query involves images, the application retrieves relevant image embeddings from the Chroma database. It then reconstructs the feature vectors which were retrieved into an image then uses an image model (Gemini) to generate context-aware responses based on the images.

Audio Input: The application also supports audio input via the OpenAI Whisper model, allowing users to speak their

queries. The audio is transcribed to text, which is then processed like regular text inputs.

Generating Responses

Agent Execution: The application employs an agent executor that utilizes various tools to gather information. The structured prompt template guides the AI's response generation, ensuring comprehensive and context-aware replies.

Response Generation: The chatbot generates a response based on the collected information. If multiple inputs (text, images, or audio) are available, it synthesizes a coherent answer that considers all relevant modalities.

Final Output

Displaying Results: The final response, whether derived from text, images, or audio, is displayed to the user in the chat interface. The application can also provide relevant context or additional resources based on the user's query. Chat History Management: All interactions are logged, allowing users to refer back to previous queries and responses, enhancing the overall user experience.

YouTube Video Summarization: The application has created a functionality to summarize YouTube videos by integrating the YouTube Transcribe API. This feature collects the transcript of any video, which is then fed into the language model to generate a summary of the video. Users can pass the URL link of the YouTube video, and the application will return a concise summary based on the transcribed content.

4. Chain-based and Agentic Execution Strategies

Chain-based and agentic execution strategies are central to the orchestration of tasks in sophisticated AI systems, including the discussed RAG application. Chain-based execution involves the sequential processing of tasks, where the output of one step serves as the input for the next. This approach is well-suited for scenarios where tasks follow a predictable flow and can be divided into discrete, interdependent stages. For instance, a chainbased execution could involve first retrieving relevant documents, then summarizing those documents, and finally generating a response based on the summary. The chainbased approach ensures that complex processes are broken down into manageable components, which can be executed and monitored step by step.

On the other hand, agentic execution strategies are more dynamic and adaptive, allowing the system to act autonomously based on the current context and the tasks at hand. In agentic execution, AI agents are capable of initiating, modifying, or terminating tasks based on their understanding of the environment and user interactions. This approach is particularly useful in scenarios where tasks are less predictable or require on-the-fly adjustments. For example, an agentic execution might involve an AI agent deciding to perform a web search in response to a user's ambiguous query, then using the search results to clarify the query before proceeding with further tasks. This adaptability makes agentic strategies suitable for handling complex, multi-step workflows that require real-time decision-making.

Combining chain-based and agentic execution strategies in a single RAG system offers significant advantages. It allows for a structured approach to task orchestration while still providing the flexibility to adapt to changing contexts. In the discussed RAG application, tasks can be organized into a series of chains while allowing agents to make autonomous decisions when deviations from the standard flow occur. This hybrid execution model improves the robustness of the system by ensuring that it can handle both predictable and unpredictable scenarios effectively. The use of frameworks such as Langchain and Langsmith further supports this by providing the tools needed for orchestrating tasks dynamically, with support for both execution approaches. Implementing these strategies requires careful design to ensure that the system can manage dependencies, error handling, and fallback mechanisms effectively. The choice between chain-based and agentic execution at any given point depends on factors such as task complexity, available data, and user input. By intelligently switching between these strategies or combining them, the RAG system can optimize task execution, reducing latency and improving the user experience. This also allows for more sophisticated AI behaviors, such as proactively retrieving additional context or adapting the response generation process based on user feedback.

5. Audio and Image Handling with OpenAI Whisper and OpenCLIP

Audio and image processing are integral parts of the discussed RAG system, expanding its capability to handle various input modalities. OpenAI's Whisper model is employed for speech-to-text conversion, allowing audio data to be processed as text and used in retrieval and generation tasks. The Whisper model is particularly well- suited for this because of its robustness in handling different accents, noise levels, and languages, ensuring accurate transcription across diverse audio inputs. Once transcribed, the audio content can be treated as text for further processing, such as summarization or contextual response generation. This capability is valuable in use cases like customer service, where phone call transcripts need to be analyzed, or in education, where lectures can be automatically transcribed and summarized.

In addition to audio handling, the system uses OpenCLIP for image processing, which involves generating embeddings that can map visual content into a shared vector space alongside text. OpenCLIP's ability to create meaningful vector representations of images allows the RAG system to cross-reference visual and textual data effectively, enabling tasks

like searching for relevant images based on textual descriptions or understanding the content of an image in the context of a conversation. This cross-modal capability is crucial for applications in fields such as e-commerce, where product searches might involve both images and descriptions, or in media analysis, where news articles and associated images need to be jointly analyzed.

Combining Whisper and OpenCLIP facilitates a truly multimodal approach, allowing the RAG system to interpret and integrate various data types simultaneously. For instance, in a scenario where a user provides an image and an audio description, the system can convert the audio to text, generate embeddings for both the text and image, and use the combined information to retrieve relevant data. This integration can improve the system's ability to generate more accurate responses by using all available modalities, providing richer and more contextually appropriate outputs.

Handling audio and image data in the RAG system also involves the use of vector databases for efficient retrieval. When audio or image embeddings are generated, they are stored in a vector database, which allows for fast nearestneighbor searches. This means that when a user query is issued, the system can quickly identify relevant audio or image content based on the similarity of their embeddings. The use of vector databases thus accelerates the processing of multimodal data and ensures that the system can handle real-time tasks efficiently, even when dealing with large datasets.

6. Contextual Awareness, Chat Memory, Callback Functions, and Vector Databases

Contextual Awareness in RAG Systems

The RAG (Retrieval-Augmented Generation) system's contextual awareness is pivotal for delivering meaningful interactions. This awareness is achieved through several key mechanisms, detailed in the table below:

Mechanism	Description	Importance
Chat Memory	The system's ability to recall and integrate previous interactions. This includes remembering past queries, user preferences, and interaction history.	Essential for coherence in ongoing conversations, particularly in customer support and tutoring, where context significantly impacts response accuracy.
Callback Functions	Programming constructs that allow the system to execute specific functions in response to user interactions. This ensures dynamic responses based on user needs.	Enhances responsiveness and interactivity, making the user experience smoother. Callback functions can also facilitate real-time updates in information retrieval.
Vector Databases	Databases that use numerical embeddings to represent content for efficient multimodal data retrieval. This technology allows for sophisticated comparisons between data points.	Enables quick similarity searches across text, audio, and images, improving retrieval efficiency for complex queries and enhancing the relevance of results returned.

The Role of Vector Databases

Vector databases are integral to the RAG system's architecture, significantly contributing to its contextual awareness by enabling efficient multimodal data retrieval. They differ from traditional databases in several key areas. Traditional databases rely on keyword-based indexing, which can be slow and may not capture semantic meaning. In contrast, vector databases utilize numerical embeddings to capture the semantic relationships between data points, allowing for faster and more intuitive searches.

Furthermore, while traditional databases primarily support text-based queries, vector databases can handle diverse data types, including text, audio, and images. This versatility makes them suitable for a wide range of applications. The query processing speed in vector databases is significantly faster, as they perform similarity searches based on the proximity of embeddings in vector space. This capability allows for efficient handling of complex queries that may combine multiple modalities, such as text, audio, and images in a single search, providing richer context and results.

Enhancing Contextual Understanding The combination of chat memory and vector databases enhances the system’s ability to maintain context over long interactions or across different types of queries. For example, if a user initially discusses a specific topic using text, the system records the context in its chat memory. If the user later provides an image or audio clip related to the same topic, the system integrates the new input with the stored context to generate a relevant and coherent response. This integration improves user engagement and satisfaction by offering a more intuitive interaction.

In educational applications, for instance, a student discussing a scientific concept could provide a related image or audio note. The system would then be able to deliver tailored responses based on the accumulated knowledge, enriching the learning experience.

Application in Education

The capabilities of contextual awareness and chat memory are particularly crucial in educational applications, enhancing various aspects of the learning experience for students. These features allow for a more personalized and effective approach to learning, addressing different needs, whether in exam preparation, homework assistance, or general study habits.

In educational settings, students often interact with a wide range of resources, including text-based notes, textbooks, multimedia presentations, and recorded lectures. For instance, when a student asks a question about a specific subject, the system can recall previous interactions and relevant materials. This includes not just text but also images, videos, and audio explanations, providing a comprehensive overview of the topic at hand.

The integration of chat memory with vector databases enables the system to track and analyze a student's progress over time. This capability is particularly beneficial in ongoing projects, research assignments, or collaborative work.

As students work through complex problems or group projects, the system can retain context from past discussions, allowing for seamless transitions between different aspects of their studies. For example, if a student is conducting research on a historical event, the system can draw from various sources—textual documents, images, and audio interviews—providing a well-rounded understanding of the topic.

Additionally, this technology supports diverse learning styles by adapting to the individual preferences of students. Some students may benefit from visual aids, while others may find verbal explanations more helpful. The system can offer tailored resources that align with each student's learning style, fostering a more engaging and effective educational experience. If a student struggles with a specific concept, the system can quickly pull relevant supplementary materials or alternative explanations based on their previous queries.

Moreover, the contextual awareness provided by chat memory allows for enhanced classroom interactions. In a group setting, teachers can reference students' prior questions or contributions, creating a more cohesive learning environment. This interaction can facilitate more in-depth discussions, helping students connect new information with what they've already learned.

Ultimately, the synergy of these technologies leads to improved educational outcomes. By providing students with personalized support, relevant resources, and a holistic view of their learning journey, the system empowers them to take control of their education, enhance their understanding of complex topics, and achieve their academic goals.

7. YouTube Summarization Using YouTube Transcript API and LLM Integration

A significant functionality of the RAG system is its ability to summarize YouTube videos by integrating the YouTube Transcript API with the language model (LLM). This feature allows the system to automatically fetch the transcript of a YouTube video and use it as input for summarization, providing users with a concise summary of the video's content. The process begins by retrieving the transcript through the YouTube Transcript API, which captures spoken content from the video and converts it into text. This text is then fed into the LLM, such as Gemini 1.5 Flash, which generates a summary that highlights the key points and main topics of the video.

The summarization feature significantly enhances the usability of the RAG system, especially in scenarios where users need quick insights into lengthy videos without watching them in full. For example, educators and students can use this feature to get quick summaries of educational videos, while professionals can use it to stay updated on relevant industry content without spending hours watching videos. By providing the YouTube video URL along with the summary, the system ensures that users can refer back to the original content if they wish to explore further. This integration also adds to the system's multimodal capabilities, bridging the gap between video content and text-based analysis.

Integrating the YouTube Transcript API into the RAG system involves handling various challenges, such as the variability in transcript quality and video content structure. The quality of transcripts can vary based on factors like video audio quality, speaker accents, and background noise. To address these issues, the system can incorporate additional preprocessing steps to clean the transcript and improve its readability before summarization. Moreover, techniques like extracting timestamps from the transcript can be used to segment the video content into meaningful sections, allowing the LLM to generate more structured and informative summaries.

The use of LLMs for summarization also provides the flexibility to tailor the output based on user requirements. For instance, the system can generate different types of summaries, such as brief overviews, detailed summaries, or even question-and-answer formats, depending on what the user prefers. This customization is enabled by finetuning the LLM on summarization tasks and incorporating prompt engineering techniques to guide the model's output. Additionally, integrating YouTube summarization into the broader RAG system opens up possibilities for further multimodal applications, such as combining video summaries with related web search results or contextual chat responses based on the video content.

By leveraging the YouTube Transcript API and LLMs, this RAG system provides a valuable tool for navigating the vast amount of video content available online. It not only saves time but also enables users to quickly identify the most relevant information, thus enhancing the overall user experience in interacting with videobased data. This capability can be extended to various domains, such as content curation, research, and knowledge management, making it a versatile addition to the RAG system's suite of functionalities.

II. METHODS AND MATERIAL

The proposed Multimodal RAG (Retrieval-Augmented Generation) system integrates various input modalities to provide comprehensive and context-aware responses through an intelligent chatbot interface. The system combines document retrieval, vector databases, and generative AI to ensure efficient and relevant information retrieval across different types of queries.

User Interaction: Users can upload various types of documents, such as PDFs, images, and CSVs, or interact with the chatbot through a user-friendly chat interface. The system also supports audio inputs and accepts YouTube URLs for summarization. It maintains the session state to track conversation history and provide continuous and coherent interactions.

File Processing: Upon receiving file uploads, the system processes these files through specialized loaders, such as PDFMinerLoader for PDFs and CSVLoader for CSVs. Images are processed and stored in a Chroma database for later retrieval when visual queries are made. This ensures that all types of files are converted into a format that can be embedded for searching.

Generating Embeddings: The system generates embeddings for text and image data using two primary technologies—Google Generative AI for text embeddings and OpenCLIP for image embeddings.

These embeddings facilitate similarity-based searches and allow the system to retrieve relevant information quickly and efficiently from the vector database.

Query Handling: When a user submits a query, the system evaluates the input and applies the appropriate tools for response generation. Text queries are processed by structured agents, which can utilize external APIs like Tavily or search the vector store for relevant data.

Imagebased queries retrieve image embeddings from the Chroma database, which are then processed by the Gemini image model for context-aware answers. Audio inputs are handled by the OpenAI Whisper model, which transcribes the audio into text for further processing.

Response Generation: The system employs an agent executor to gather the required information, whether it be from text, images, or audio. A structured prompt template guides the AI's response generation, ensuring that the response is comprehensive and contextually appropriate. In cases where multiple input modalities are used, the system synthesizes all relevant information to generate a coherent and accurate response.

YouTube Video Summarization: The system includes functionality for summarizing YouTube videos by integrating the YouTube Transcribe API. Users can provide a video URL, and the system collects the video transcript. The transcript is then fed into the language model, which generates a concise summary of the video content based on the transcribed information.

Final Output: The final response, whether it is based on text, images, or audio, is displayed to the user in the chat interface. If a YouTube URL is provided, the summarized content is presented in the same chat interface. The system also maintains chat history for users to refer back to previous queries and responses, improving the overall user experience.
Workflow

User Interaction: Users upload documents, type queries, or submit audio inputs or YouTube URLs for summarization. The system maintains conversation history to ensure continuity across interactions.

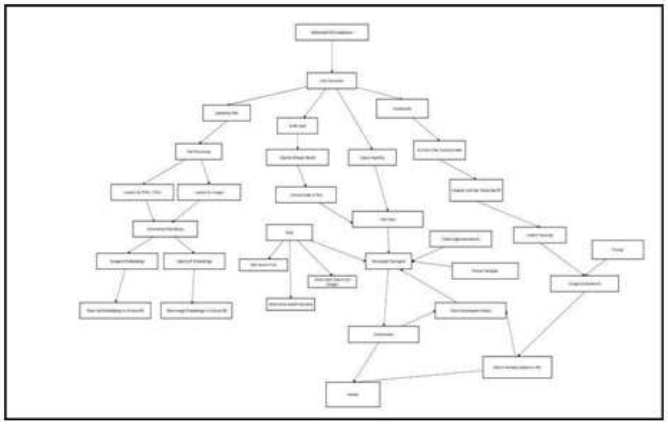
Processing Inputs: The application processes the uploaded files, converting them into embeddings that facilitate efficient searching and retrieval. Text data is embedded using Google AI embeddings, while image data is embedded using OpenCLIP and stored in the Chroma database for future visual retrieval.

Query Handling: User queries, whether text, image, or audio, are evaluated, and the system utilizes the appropriate tools to generate responses. For image queries, embeddings from the Chroma database are used to generate a context-aware response, while audio input is transcribed and processed like text.

Response Generation: The system employs an agent executor that collects the necessary information from the various modalities and tools, guided by a structured prompt template to ensure the response is complete and relevant.

Final Output: The generated response is presented to the user, and if a YouTube URL was provided, a concise summary

of the video content is displayed. All interactions are logged for future reference.



III. CHALLENGES AND LIMITATIONS

This project, despite its innovative approach using Google's Gemini API and multimodal RAG-based methods, encounters certain challenges and limitations.

1. Complexity of Multimodal Data Integration
Integrating different data types (text, video, images, and structured documents like resumes) presents challenges, as each data type requires distinct preprocessing and handling methods. Maintaining consistency in processing and interpreting diverse formats is complex, which can increase computational costs and processing time.

2. Dependency on External APIs
The application's reliance on external APIs, like Google's Gemini and YouTube Data API, poses a limitation. Any change, rate limit, or downtime in these APIs could disrupt the application's functionality. Additionally, privacy and data-sharing limitations with third-party APIs might restrict access to certain data or features.

3. Scalability Issues
Handling and processing large volumes of multimodal data can be resource-intensive, especially if deployed on limited infrastructure. As user demand grows, maintaining real-time responses for video summarization, text processing, and resume parsing may become a bottleneck, requiring optimization for scalability.

4. Latency and Response Time
Multimodal RAG applications require extensive data retrieval, processing, and generation steps, potentially leading to higher latency. Real-time applications, especially those interacting with video content, may struggle with response delays, impacting user experience.

5. Interpretability and Transparency
Multimodal AI models, especially those using complex APIs and RAG frameworks, often operate as black boxes, making it difficult to interpret how they generate responses. This lack of transparency could be a challenge for users needing clear reasoning behind AI- generated insights, especially in fields like recruitment where decision explanations are crucial.

IV. CONCLUSION

The Advanced Multimodal RAG application represents a substantial advancement in context- aware, multimodal AI systems by combining innovative language modeling, multimodal embeddings, and robust task orchestration frameworks. The integration of chain-based and agentic execution models enables the system to handle both structured workflows and adaptive, real- time decision-making. Key features, such as YouTube video summarization, audio processing, and multimodal embedding-based retrieval, extend the system's applicability across various domains, including education, customer service, and knowledge management.

While the system demonstrates impressive multimodal capabilities, challenges like API dependency and latency must be addressed to ensure scalability and optimal performance in real-time applications. This project emphasizes the importance of modular, flexible architecture in building advanced RAG applications, pointing to promising directions for future developments in multimodal AI systems

REFERENCES

1. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks R. S. Sutton and A. G. Barto, Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, Computation and Language, 2014, 2015.
2. PythonLangChain PythonLangChain. [Online]
3. Reinforcement Learning : An Introduction R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
4. Robust Speech Recognition via Large-Scale Weak Supervision A Baevski et al., "Robust Speech Recognition via Large-Scale Weak Supervision," Proc. of the International Conference on Learning Representations (ICLR), 2021.
5. Learning Transferable Visual Models From Natural Language Supervision A. Radford, J. W. Kim, T. Salimans, et al., "Learning Transferable Visual Models From Natural Language Supervision," in International Conference on Machine Learning (ICML), 2021.
6. What is the Metaverse? An Immersive Cyberspace and Open Challenges A. Mystakidis, "What is the Metaverse?

An Immersive Cyberspace and Open Challenges," IEEE Access, vol. 10, pp. 622–629, 2022.
7. Scaling Laws for Neural Language Models J. Kaplan, S. McCandlish, T. Henighan, et al., "Scaling Laws for Neural Language Models," arXiv preprint, arXiv:2001.08361, 2020
8. Chroma DB Chroma DB. [Online] Available: https://github.com/chroma-db/chroma