Project Report On

Leveraging LLMs with RAG for Q&A systems

Submitted By
DHAIRYA THAKKAR 22011097
DHIREN OSWAL 22011106
SOHAM TOLWALA 22010224
PUSHPAK SURYAWANSHI 22010400

Under the Guidance of PROF SWAPNIL SHINDE

In Partial fulfillment of
Bachelor of Technology
Artificial Intelligence and Data Science
2023-2024
At

Department of Artificial Intelligence and Data Science
Vishwakarma Institute of Information Technology, Pune 411048

Affiliated To

Savitribai Phule Pune University, Pune Bansilal Ramnath Agarwal Charitable Trust's

Certificate

This is to certify that the project report entitles

“ Leveraging LLMs with RAG for Q&A systems”

Submitted by

DHAIRYA THAKKAR	**Exam No : 22011097**
DHIREN OSWAL	**Exam No : 22011106**
SOHAM TOLWALA	**Exam No : 22010224**
PUSHPAK SURYAWANSHI	**Exam No : 220110400**

is a bonafide work in partial fulfillment of the award of Bachelor of Technology in Artificial Intelligence and Data Science, Savitribai Phule Pune University, Pune, during the year 2023-24. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Technology Degree.

Prof. Swapnil Shinde
Project Guide

Prof(Dr.) P. N.Mahalle
Head, AInDS Department

Prof.(Dr.)Vivek. Deshpande
Director, VIIT, Pune

Date:

Examiners: 1. . . . . 2. . . . .

Place: Pune

Acknowledgement

We take this opportunity to thank Head of the Department **Dr. P.N.Mahalle** and our project guide **Prof. Swapnil Shinde** for their valuable guidance and providing all the necessary facilities, which were indispensable in the completion of this project report. We are also thankful to all the staff members of the Department of Artificial Intelligence and Data Science, VIIT, Pune for their valuable time, support, comments, suggestions and persuasion.

DHAIRYA THAKKAR 22011097
DHIREN OSWAL 22011106
SOHAM TOLWALA 22010224
PUSHPAK SURYAWANSHI 22010400

ABSTRACT

Large Language Models (LLMs) have shown potential in various domains but can struggle with factual accuracy and knowledge gaps because of unseen or outdated data. Retrieval-Augmented-Generation (RAG) offers a solution by integrating information retrieval from reliable sources with LLM generation, leading to improved performance in knowledge-intensive domains. This paper explores the application of LLM+RAG for question-answering in specialized domains. We discuss the state-of-the-art RAG framework, including retrieval, generation, and augmentation techniques. To tailor the needs of specialized domain inquiries, exemplified by its novel application in the National Eligibility-cum-Entrance Test (NEET), a standard Indian medical entrance exam for medical education, the aim is to furnish precise and contextually relevant answers. We utilized two LLM models, Gemini-pro and gpt-3.5-turbo-0125, in conjunction with GoogleGenerativeAIEmbeddings embeddings and Chroma vector DB. The National Council of Educational Research and Training (NCERT) textbooks were fed as the data sources to extend the knowledge base of LLM. The questions were taken from the 2022 NEET paper and the RAG-based systems were evaluated with the ground truth answers given along the paper in which the GPT incorporated model scored the highest with an overall 62.5% precision. Through a comprehensive examination of state-of-the-art technologies and frameworks, we elucidate the potentials and drawbacks of LLMs with RAG and jot down the necessary checkpoints in advancing question-answering systems across specialized domains.

Keywords—Question-Answering Systems, Large Language Models, Retrieval Augmented Generation Approach, Information Retrieval and Generation

Index

LIST OF FIGURES

LIST OF TABLES

ABBREVIATION ILLUSTRATION

RAG	Retrieval Augmented Generation
LLM	Large Language Model

Chapter 1

Introduction

1.1 Motivation

Motivated by the desire to explore the capabilities of the Retrieval-Augmented Generation (RAG) framework in conjunction with Large Language Models (LLMs) in specialized domain question answering, our focus turns to addressing unexplored problem statements. One such example is the application of this LLM-RAG combination to solve India's National Eligibility cum Entrance Test (NEET), a critical examination for medical admissions. While NEET presents a complex and specialized domain, there has been minimal research into utilizing advanced NLP techniques to aid in its preparation or assessment. Hence, our motivation lies in pioneering the use of LLM-RAG in this domain to assess its effectiveness in providing accurate and comprehensive answers to medical-related questions.

This exploration into leveraging LLM-RAG for NEET presents an opportunity to assess the adaptability and robustness of these advanced NLP techniques in a domain characterized by intricate medical concepts and terminology. By applying this combination to NEET preparation, we aim to investigate its ability to retrieve relevant information from vast medical knowledge repositories, generate coherent and contextually accurate responses, and ultimately aid students in their exam readiness.

Furthermore, the motivation behind this endeavor extends beyond NEET to showcase the potential of LLM-RAG in addressing specialized domain question answering across various fields such as medicine, law, and engineering. Through rigorous experimentation and evaluation, we seek to contribute valuable insights into the applicability and limitations of this approach, paving the way for future research and development in utilizing advanced NLP techniques for specialized domain tasks.

1.2 Background

In recent years, Large Language Models (LLMs) have emerged as powerful tools in natural language processing, showcasing remarkable capabilities across various domains. Models like Mistral, OpenAI's GPT series, Google's Gemini (previously known as Bert), and Meta’s Llama have demonstrated proficiency not only in general language understanding but also in specialized fields such as medicine, finance, and business. OpenAI's ChatGPT, supported by Microsoft, stands out for its flexibility, seamlessly transitioning between tasks like coding and music composition, and achieving notable success in academia, including passing the law exam at Minnesota Law School and clearing the United States Medical Licensing Examination (USMLE) in one attempt with 60% accuracy. However, despite these achievements, the performance of LLMs in specialized domain tasks remains a subject of scrutiny, particularly in complex examinations like the National Eligibility cum Entrance Test (NEET) for medical admissions.

The NEET examination presents unique challenges, encompassing questions from physics, chemistry, and predominantly biology. Success in NEET not only requires factual knowledge but also demands the ability to comprehend complex concepts and apply them in problem-solving scenarios. Traditional LLMs have struggled to meet these requirements, facing issues such as hallucination, outdated information, and opaque reasoning processes, which can compromise their accuracy and reliability. Consequently, there is a pressing need to evaluate the suitability of LLMs like ChatGPT in addressing the specific demands of NEET and other specialized domain tasks.

1.3 Brief Introduction

OpenAI's ChatGPT, supported by Microsoft, is famous for its flexibility. It can easily switch between tasks like creating complex codes and composing music. Its abilities aren't limited to just technical or literary tasks; it has also achieved significant success in academics. For instance in specialized domains, it performed exceptionally well in the MBA program at the University of Pennsylvania and passed the law exam at Minnesota Law School[4]. Moreover, it successfully cleared the United States Medical Licensing Examination (USMLE)[5] in one with 60% accuracy. Normally, aspiring doctors spend nearly four years in medical school and over two years in clinical rotations to pass this exam[6].

But, after a failed attempt at the Union Public Service Commission (UPSC) exam[7], AIM decided to check ChatGPT's medical prowess[8]. The NEET examination, a crucial milestone for aspiring medical professionals, presents a unique set of challenges. With questions spanning diverse topics in physics, chemistry, and majorly in biology, NEET demands not only factual knowledge but also the ability to comprehend complex concepts and apply them in problem-solving scenarios. So, when ChatGPT took the NEET 2022 exam with 174 out of 200 questions, it just passed, scoring 50.14% (357 out of 712), while struggling particularly in biology[8]. It proves that traditional LLMs often struggle to meet these requirements, facing issues like hallucination, outdated information, and opaque reasoning processes, which can impede their accuracy and reliability [9], [10].

In response to these challenges, Retrieval-Augmented Generation (RAG) has emerged as a promising approach to enhance the capabilities of LLMs in specialized domains [11], [12], [13], [14]. By integrating knowledge from external sources through retrieval mechanisms, RAG enables LLMs to generate contextually relevant responses grounded in credible information retrieved from an extended curated data source. This paper explores the maiden utilization of RAG in optimizing question-answering systems for specialized domains like NEET, with a focus on assessing the accuracy and potential to tailor responses to complex inquiries. Furthermore, it is noteworthy that when ChatGPT took the NEET exam, it achieved nearly a 50% accuracy rate with low scores, especially in Biology[8]. However, by leveraging RAG techniques, the QA system scored with an overall accuracy rate of 61.1%, and a precision rate of 62.5% on the same exam when evaluated against the original answers. In addition to highlighting the efficacy of RAG in enhancing LLMs' performance in specialized domains like NEET, this paper explores the analysis from the score obtained from GPT and Gemini to help in navigating future developments for specialized QA domains.

By leveraging user queries, document retrieval, context integration, and RAG-enabled response generation, our approach seeks to address the unique challenges posed by specialized domains like NEET, ultimately contributing to the development of more accurate and reliable question-answering systems for educational and professional purposes. We primarily assess the models using basic metrics like precision and accuracy. However, we also provide an overview of evaluation methodologies/frameworks for a more comprehensive understandin

Chapter 2

Literature Survey

1.1 Literature Review

Title	**Author**	**Year**	**Methodology**	**Results**	**Gaps**
Retrieval-Augmented Generation Paradigms	[11]	2023	Survey of RAG frameworks like Naive RAG, Advanced RAG, and Modular RAG, focusing on core components and evaluation methods	Emphasizes evaluation methodologies, including quality scores such as context relevance, answer faithfulness, and answer relevance	Identifies challenges in evaluating LLM performance and highlights the necessity of comprehensive evaluation frameworks
Addressing Challenges in Educational Math Q&A with RAG	[14]	2022	Implementation of RAG system for Math chatbot, utilizing retrieval and generation for improved accuracy	Demonstrates RAG's effectiveness in mitigating LLM shortcomings and enhancing answer interpretability	Identifies challenges specific to educational Math Q&A and the need for clear explanations and step-by-step solutions
Optimization Strategies for RAG-based Chatbots	[16]	2023	Exploration of Reinforcement Learning (RL) to optimize RAG-based chatbots, reducing resource consumption	Achieves resource consumption reduction compared to traditional RAG	Suggests exploring diverse optimization strategies for further improvements

				approaches	
ACTIVERAG : Active Learning for RAG Systems	[17]	2022	Introduction of active learning approach for RAG systems, identifying most informative context for each question	Surpasses previous RAG models, achieving a 5% improvement on question-answering datasets	Highlights the potential of active learning in enhancing RAG system efficiency and accuracy
Resources for RAG Systems in Medical QA	[18]	2023	Introduction of MIRAGE benchmark and MedRAG toolkit for medical QA	Provides resources for developing and evaluating effective RAG systems for medical QA	Indicates the need for specialized resources and evaluation benchmarks for medical QA
PaperQA: Agent-based Architecture for Scientific QA	[19]	2022	Introduction of PaperQA architecture for scientific QA tasks	Introduces LitQA dataset for PaperQA evaluation, demonstrating improved accuracy and contextually grounded answers	Highlights the potential of PaperQA architecture in scientific QA tasks and the need for evaluation datasets

Applications of RAG Beyond QA Systems	[20]	2023	Exploration of RAG applications beyond QA systems, focusing on claim verification	Identifies RAG's potential for claim verification, where retrieving relevant evidence is crucial for determining claim veracity	Indicates the broader applicability of RAG beyond QA systems and suggests further exploration in diverse domains

1.2 Review of existing System

Based on the review of existing literature, it is evident that Retrieval-Augmented Generation (RAG) has emerged as a promising approach to enhance the capabilities of Large Language Models (LLMs) for question-answering (QA) systems across various domains. Existing systems have explored different aspects of RAG, ranging from surveying frameworks and methodologies to implementing optimization strategies and exploring applications in specialized domains like education and healthcare.

Existing systems have identified several challenges faced by traditional LLM-based QA systems, such as hallucinations, limited interpretability, and struggles with nuanced concepts. These challenges can be particularly pronounced in critical domains like education and healthcare, where accurate and reliable information is paramount. In response to these challenges, RAG techniques have been proposed and implemented to augment LLMs with external knowledge sources through retrieval mechanisms.

Several studies have focused on evaluating the effectiveness of RAG in enhancing LLM-based QA systems. For example, research by [11] provides a comprehensive survey of RAG frameworks, outlining their core components and evaluation methods. This study emphasizes the importance of evaluation methodologies, particularly in assessing both retrieval and generation components of RAG systems. Similarly, [14] demonstrates the effectiveness of RAG in mitigating LLM shortcomings and enhancing answer interpretability, particularly in educational Math QA contexts.

Optimization strategies for RAG-based chatbots have also been explored, with studies like [16] investigating the use of Reinforcement Learning (RL) to optimize resource consumption. Additionally, [17] introduces active learning approaches for RAG systems, identifying the most informative context for each question and achieving improvements in efficiency and accuracy.

In specialized domains like medical QA, resources like the MIRAGE benchmark and MedRAG toolkit have been developed to facilitate the development and evaluation of effective RAG systems [18]. These resources provide valuable insights and tools for researchers and developers working with RAG systems in the medical domain. Furthermore, applications of RAG extend beyond traditional QA systems, with studies like [19] exploring its potential in scientific QA tasks and [20] investigating its applicability in claim verification. These studies highlight the versatility of RAG in addressing a wide range of information retrieval and generation tasks across diverse domains.

In the context of our system, we leverage the insights and methodologies derived from existing literature to develop an optimized RAG-based QA system tailored for specialized domains like NEET preparation. By integrating user queries, document retrieval, context integration, and RAG-enabled response generation, our system addresses the unique challenges posed by NEET and aims to provide accurate and reliable answers to complex inquiries. Additionally, we explore optimization strategies and evaluation frameworks outlined in the literature to enhance the performance and effectiveness of our system in educational and professional contexts.

Chapter 3

Project Statement

3.1 Purpose behind the Project

The purpose of the project is to leverage Large Language Models (LLMs) integrated with Retrieval-Augmented-Generation (RAG) techniques for question-answering in specialized domains, focusing specifically on the National Eligibility-cum-Entrance Test (NEET), an Indian medical entrance exam. The goal is to enhance the accuracy and contextuality of answers provided by LLMs by incorporating information retrieval from reliable sources. This project aims to address knowledge gaps and improve performance in knowledge-intensive domains like medical education.

Key objectives of the project include:

1. Implementing state-of-the-art RAG framework: The project aims to explore and implement advanced techniques in retrieval, generation, and augmentation to enhance the question-answering capabilities of LLMs.
2. Customizing for specialized domain inquiries: Tailoring the system to meet the specific needs of NEET questions, including precision, context relevance, and factual accuracy.
3. Utilizing multiple LLM models and embeddings: Leveraging LLM models such as gemini-pro and gpt-3.5-turbo-0125, along with GoogleGenerativeAIEmbeddings embeddings, to improve the quality of generated answers.
4. Extending knowledge base with relevant data: Incorporating data from National Council of Educational Research and Training (NCERT) textbooks to enrich the knowledge base of LLMs and improve their understanding of medical concepts.
5. Evaluating performance against ground truth: Validating the accuracy of generated answers by comparing them with the ground truth answers from the 2022 NEET paper, ensuring the system's reliability and effectiveness.

3.2 Decision of Scope

In determining the scope of this research project, several key factors were considered to ensure a comprehensive and focused exploration of the integration of Large Language Models (LLMs) with Retrieval-Augmented-Generation (RAG) techniques for question-answering in specialized domains, specifically within the context of the National Eligibility-cum-Entrance Test (NEET) for medical education in India.

1. Domain Specificity: The scope of the project is tightly aligned with the specialized domain of medical education, as exemplified by the NEET exam. This focus allows for a deep dive into the intricacies of medical terminology, concepts, and contextual understanding required for accurate question-answering, catering directly to the needs of students and professionals in this domain.

2. Technology Integration: The project's scope includes the integration and utilization of cutting-edge technologies such as LLM models like gemini-pro and gpt-3.5-turbo-0125, as well as advanced embeddings such as GoogleGenerativeAIEmbeddings and the Chroma vector DB. This integration ensures the development of a robust and technologically advanced question-answering system capable of handling complex queries with high accuracy.

3. Data Source and Knowledge Base Expansion: The scope encompasses the use of data from National Council of Educational Research and Training (NCERT) textbooks, a reputable and authoritative source in the Indian education system. This data serves as the foundation for expanding the knowledge base of LLM models, enabling the system to provide accurate and contextually relevant answers to NEET-related queries.

4. Performance Evaluation: Within the scope of the project, a rigorous evaluation process is undertaken to assess the performance of the LLM+RAG question-answering system. This evaluation involves comparing the system-generated answers to ground truth answers extracted from the 2022 NEET paper, ensuring the system's reliability, accuracy, and ability to handle real-world exam questions.

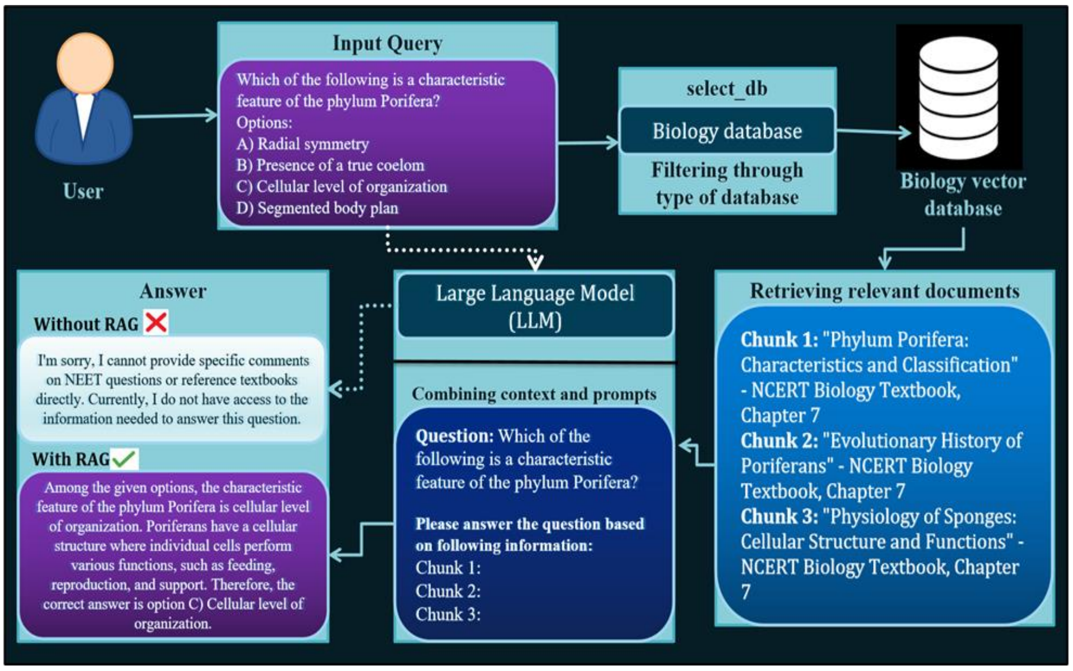
5. Contextual Relevance: An essential aspect of the project's scope is to ensure that the generated answers are not only accurate but also contextually relevant within the NEET exam framework. This includes considering the specific question formats, topics, and knowledge domains covered in the exam to provide meaningful and valuable responses to users.

By defining a clear and focused scope, the project aims to contribute valuable insights and advancements to the field of question-answering systems, particularly in specialized domains like medical education, showcasing the potential of LLM+RAG integration to address complex queries and enhance user experiences.

3.3 Methodology for solving this proposed theme

3.3.1 Proposed system Architecture.

The Retrieval-Augmented Generation (RAG) workflow is a multi-step process designed to facilitate efficient information retrieval and response generation. Initially, it involves preprocessing a substantial corpus of relevant text data, such as textbooks and study materials, by breaking it down into smaller, more manageable pieces. These pieces are then encoded into a format optimized for quick and effective searching. When a user query is received, the RAG system's retrieval component comes into play. This retrieval agent is responsible for searching through the encoded data to identify segments that closely match the content and context of the query. This search is conducted based on various factors, including semantic relevance and keyword associations. Once relevant segments of text are identified, the RAG system utilizes this information to craft a response to the user query. By leveraging the retrieved context, the system can generate responses that are not only accurate but also contextually relevant and well-informed. Overall, the RAG workflow enables the system to dynamically retrieve and integrate relevant information from the corpus, empowering it to provide comprehensive and accurate responses to user queries. In the subsequent sections, we will delve further into the specific architectural framework and implementation details of the RAG system within our NEET QA application. Figure 1 gives the architecture of the system.


User
Input Query
select_db
Biology database
Filtering through type of database
Biology vector database
Retrieving relevant documents
Large Language Model (LLM)
Combining context and prompts
Answer
Without RAG ❌
With RAG ✓

Which of the following is a characteristic feature of the phylum Porifera?
Options:
A) Radial symmetry
B) Presence of a true coelom
C) Cellular level of organization
D) Segmented body plan

Chunk 1: "Phylum Porifera: Characteristics and Classification" - NCERT Biology Textbook, Chapter 7
Chunk 2: "Evolutionary History of Poriferans" - NCERT Biology Textbook, Chapter 7
Chunk 3: "Physiology of Sponges: Cellular Structure and Functions" - NCERT Biology Textbook, Chapter 7

Question: Which of the following is a characteristic feature of the phylum Porifera?

Please answer the question based on following information:
Chunk 1:
Chunk 2:
Chunk 3:

I'm sorry, I cannot provide specific comments on NEET questions or reference textbooks directly. Currently, I do not have access to the information needed to answer this question.

Among the given options, the characteristic feature of the phylum Porifera is cellular level of organization. Poriferans have a cellular structure where individual cells perform various functions, such as feeding, reproduction, and support. Therefore, the correct answer is option C) Cellular level of organization.

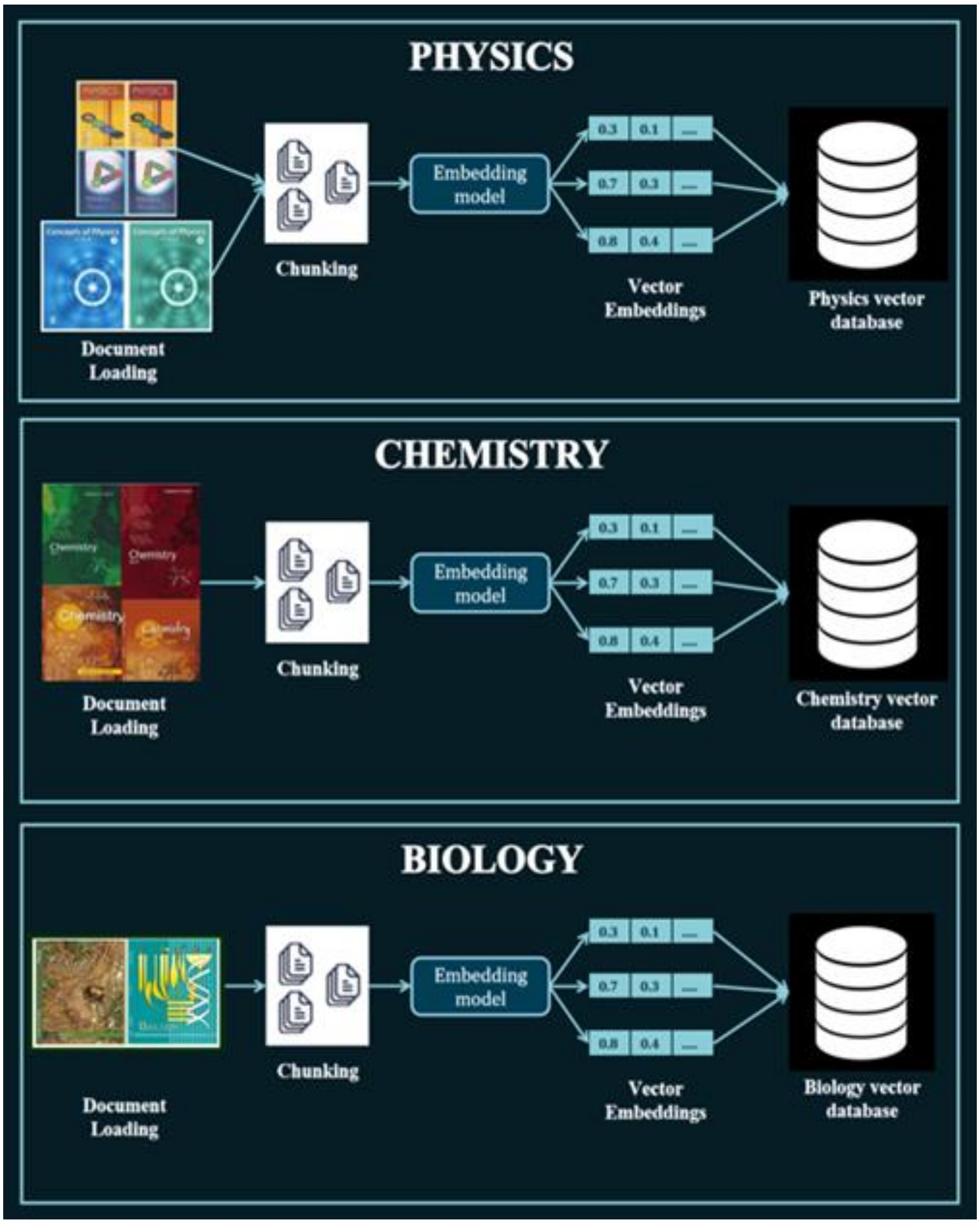
Figure 1 Architecture

A. User Input:

The initial step in the Retrieval-Augmented Generation (RAG) workflow involves the user input, which is typically a query derived from the NEET question paper. Our approach began with the utilization of the NEET 2022 Question set as the primary source for extracting questions. These questions were meticulously chosen to ensure their relevance to the NEET exam, thereby aligning with the specific requirements of our research. However, it was imperative to exclude questions containing complex formatted LaTeX text, diagrams, or images from our dataset. This decision stemmed from the limitation of the rudimentary model we employed, which could only process textual data and not handle more complex equation representations or visual content. Once the relevant questions were identified, they were amalgamated with their corresponding ground truth answers, organized efficiently in a JSON format. Each question-answer pair served as a distinct entry in our dataset, facilitating systematic data management and analysis. For instance, a typical question from the dataset might be: "Which of the following is a characteristic feature of the phylum Porifera? Options: A) Radial symmetry B) Presence of a true coelom C) Cellular level of organization D) Segmented body plan". While ChatGPT serves as a widely utilized Large Language Model (LLM), its knowledge is primarily derived from pretraining data, which may not encompass the latest information or specialized domain knowledge required for the NEET exam. To address this limitation, we leveraged the RAG framework to incorporate up-to-date document excerpts from reputable external knowledge repositories. These included NCERT textbooks for physics, chemistry, and biology, Oswal Revision notes for biology, and Concepts of Physics by Prof. H.C. Verma for physics. These resources are highly regarded among NEET aspirants for their comprehensive coverage of relevant study materials. By amalgamating questions from the NEET 2022 Question Set with pertinent external resources, we curated a comprehensive dataset that encapsulated all study materials essential for NEET exam preparation. This dataset served as the cornerstone for evaluating our RAG model's performance, enabling it to generate well-informed responses to user queries derived from the NEET question paper. The incorporation of external knowledge sources through the RAG framework augmented the LLM's capabilities, enriching the prompt with relevant contextual information and enhancing the quality of generated responses. This underscores the effectiveness of RAG in bolstering the performance of LLMs by integrating external knowledge sources tailored to the specific requirements of the NEET exam.

B. Creating Vector Database

The creation of the vector database marks a pivotal stage in the preparatory phase of our operations, laying the foundation for subsequent analysis. This process encompasses several essential phases aimed at transforming raw data into a structured format conducive to effective analysis. Initially, the data undergoes rigorous cleansing and extraction procedures to ensure its integrity and consistency. Various file formats commonly used for storing or presenting content pertinent to the specialized domain, such as PDF, HTML, Word, and Markdown, are standardized into plain text, thereby facilitating seamless integration and analysis. To enrich our database with relevant content, we obtained NCERT e-books from [21] and meticulously parsed them to extract the textual information from the textbooks. This meticulous extraction process ensures that our database is populated with comprehensive and up-to-date content relevant to the NEET exam syllabus. To address the inherent limitations of language models related to context windows, the extracted text undergoes segmentation into smaller, more manageable chunks through a process known as chunking. This segmentation can be performed based on various criteria, including character limits, specific titles, or other contextual markers, enabling more focused analysis and retrieval. Subsequently, these segmented text chunks are transformed into vector representations using an embedding model carefully selected for its optimal balance between inference efficiency and model size. In our approach, we employ the GoogleGenerativeAIEmbeddings embeddings, known for their robust performance and versatility in capturing semantic information. These embeddings serve as numerical representations of the textual content, encapsulating its semantic meaning and contextual nuances. Finally, the vector representations are stored in a dedicated database, referred to in this context as a vector store. This vector store serves as a repository for the vector embeddings, allowing for efficient retrieval and manipulation during the subsequent stages of our operations. By organizing the data in this structured manner, we ensure accessibility, scalability, and ease of integration with the retrieval and generation components of our RAG framework, thereby facilitating seamless operation and enhancing the overall effectiveness of our system.When text is stored in a vector database like ChromaDB, it undergoes a crucial transformation where textual information is encoded into numerical vector formats within a high-dimensional space. This encoding process is essential as it enables efficient similarity comparisons during the retrieval phase, a functionality facilitated by tools such as Chroma vector store’s similarity search mechanism. To ensure comprehensive retrieval capabilities and facilitate further analysis, we also store the source of the document as metadata within the vector database. This metadata serves as valuable contextual information, enhancing the interpretability and relevance of the retrieved document chunks. In the construction of our vector database, we adhere to a structured approach that optimizes both efficiency and scalability. The default schema for storing documents in the vector database is meticulously designed to accommodate various document types while maintaining consistency and integrity. By adhering to this standardized schema, we ensure uniformity and ease of access across the entire database.



Furthermore, we strategically incorporate three distinct vector databases, each dedicated to a specific subject area, as illustrated in Figure II. This strategic division allows for focused retrieval and analysis within each subject domain, thereby enhancing the accuracy and relevance of the retrieved document chunks. Additionally, by segregating the vector databases based on subject areas, we streamline the retrieval process and minimize computational overhead, ultimately optimizing the overall performance of our system.Overall, the creation of the vector database represents a critical step in the development of our Retrieval-Augmented Generation (RAG) framework, providing the foundational infrastructure necessary for efficient and scalable search capabilities. Through careful design and meticulous implementation, we ensure that our vector database meets the diverse needs of our system while enabling seamless integration with the retrieval and generation components of the RAG framework. This approach is meticulously crafted to alleviate the strain on computational resources, thereby elevating the efficiency and overall performance of the system. With an average of approximately twenty-five PDF documents per subject, each spanning 20 pages, the sheer volume of data underscores the intricate and extensive nature of the information associated with each subject domain. This abundance of data necessitates a robust and streamlined approach to processing and managing it effectively.

In the subsequent sections, we will delve into the methodologies employed to handle this considerable volume of data efficiently. The entire process of creating the vector database, as depicted in Figure II, serves as the cornerstone for enabling efficient retrieval and processing of information essential for tasks such as Q&A queries in large document repositories or libraries, particularly within the framework of Retrieval-Augmented Generation (RAG).

Through the integration of innovative approaches such as context-aware chunking and leveraging tools like DocumentLoader and TextSplitter, we achieve meaningful segmentation of documents. This segmentation plays a pivotal role in enhancing the efficacy of RAG applications by enabling precise retrieval and analysis of relevant information. By adopting these advanced techniques, we aim to optimize the performance of our system and ensure seamless integration with the RAG framework, ultimately facilitating more accurate and contextually grounded responses to user queries.

C. Retrieving relevant documents

Upon receiving a user query, our system meticulously initiates the retrieval process, leveraging the encoding model utilized during the indexing phase to seamlessly transform the query into a comprehensive vector representation. This vectorized query serves as the cornerstone for the subsequent exploration of the indexed corpus.

Utilizing Chroma vector store’s advanced similarity search mechanism, our system meticulously compares the vectorized query against the extensive repository of vectorized chunks within the corpus. By meticulously calculating similarity scores between the query vector and the vectorized chunks, our system identifies and selects the top K chunks that exhibit the highest degree of similarity to the query.

These carefully curated chunks serve as an expansive contextual framework, meticulously expanding upon the user's initial query to encompass a wealth of relevant information extracted from the indexed corpus. This meticulous process ensures that the synthesized response is deeply rooted in pertinent and contextually relevant information, thus elevating the overall effectiveness and accuracy of the Retrieval-Augmented Generation (RAG) framework employed by our system.

Through this comprehensive approach, our system endeavors to not only provide users with accurate and well-informed responses but also to enhance their overall understanding and engagement with the queried topics. By leveraging sophisticated retrieval mechanisms and advanced encoding models, our system aims to deliver a seamless and enriching user experience, facilitating enhanced learning and knowledge acquisition.

D. Answer Generation

In our meticulously crafted methodology, the answer generation process unfolds through a series of meticulously orchestrated steps, each designed to uphold the standards of accuracy and relevance paramount to our objectives.To commence this intricate process, we turn to the NEET 2022 Q1 question set, carefully selecting questions that encapsulate the breadth and depth of knowledge requisite for the examination. These questions undergo a meticulous journey, iteratively processed through a prompt instance, where each query serves as a catalyst for subsequent actions.

Upon receiving a query, our system springs into action, engaging both the Google Gemini-pro and GPT-3.5-turbo-0125 large language models (LLMs) sequentially. Each model is tasked with unraveling the intricacies of the query, embarking on a journey to distill its essence and extract the relevant insights embedded within.

The query, imbued with the essence of inquiry, undergoes a transformative journey. Initially embedded within the fabric of the model's understanding, it is then subjected to a rigorous vector search operation, traversing the depths of our meticulously curated corpus to retrieve the most salient documents. With precision as our guiding principle, we set K to 1, ensuring that only the most pertinent document is brought forth to illuminate the path ahead.

Once the context from the retrieved document comes to light, it merges seamlessly with the original prompt template, tailored to the specific subject area under scrutiny. Whether delving into the intricacies of biology or unraveling the mysteries of physics, our system ensures that the synthesized response is imbued with the richness of context and relevance, poised to enlighten and inform.

```
prompt_template_bio = """

You are a question answering bot which is specialized in BIOLOGY. Answer the
following Multiple Choice based question which has one single correct answer from
the four options given in the question itself. you have to accurately choose the
correct option and return it.

Choose the answer according to the context provided below.



GOLDEN RULE: You will be evaluated by your predicted option and hence focus on
returning only the option.



Context: {context}

Question: {question}



Don't answer if the context provided to you is irrelevant, if you are not sure and
confident and decline to answer and say "Sorry, I don't have much information about
it."

Include a brief reason for why you chose a particular option.

Answer:

"""
```

This prompt template serves as a guiding framework for the question-answering bot, specifically tailored to the domain of biology. It delineates a set of instructions and guidelines for the bot to follow when presented with a multiple-choice question pertaining to biology. The template begins by establishing the bot's identity as a specialized question-answering entity proficient in biology. It outlines the task at hand: to accurately select the correct answer from the provided multiple-choice options and return it to the user. Emphasizing the importance of precision, the template highlights that the bot will be evaluated based on its predicted option, underscoring the significance of selecting the correct answer. The context section within the template serves as a placeholder for the relevant background information or context associated with the question. This context is essential for the bot to comprehend the question and provide an informed response. Following the context section, the template includes placeholders for the question itself, ensuring that the bot receives the query in a structured format. This enables the bot to parse and understand the question effectively. The template also incorporates instructions for the bot to handle situations where the provided context is irrelevant or if the bot is unsure or lacks confidence in its response. In such cases, the bot is instructed to decline to answer and provide a polite refusal, stating "Sorry, I don't have much information about it." Finally, the template includes a section for the bot to formulate its response. Here, the bot is expected to provide the selected answer along with a brief reason or rationale for why it chose that particular option. This allows the bot to justify its decision and enhance transparency in its response process. The integration of the combined prompt, enriched with contextual information, marked a pivotal step in the answer generation process. This amalgamated prompt served as the cornerstone for the model's response generation, harnessing the synergistic power of both the prompt's inherent structure and the additional contextual insights retrieved from the document repository. In instances where the model encountered difficulties in retrieving an answer or responded with a lack of information, signified by the phrase "Sorry, I don't have much information about it," a fallback strategy was swiftly enacted. This strategy involved transitioning to the RAG-free mode, allowing the model to rely solely on its internal knowledge base to generate a response. This adaptive approach ensured a comprehensive exploration of potential answers, guaranteeing that each question underwent thorough processing and received a response that was as contextually relevant and accurate as possible.

Furthermore, the challenges encountered during the initial testing phase with the GPT-3.5-turbo-0125 model served as a catalyst for the adoption of RAG within our framework. These challenges, including restricted access to specialized domain knowledge and the model's struggles in interpreting complex queries, underscored the critical importance of leveraging external knowledge sources through the RAG framework to augment the model's performance. By incorporating RAG, we addressed these limitations head-on, empowering the model with access to a wealth of up-to-date and domain-specific information, thereby enhancing its ability to generate accurate and informed responses.

3.3.2 Working of the System

1. User Input: The process begins with the user inputting a query or question related to the NEET exam.
2. Extract Relevant Questions: The system extracts relevant questions from the NEET exam question set. Questions containing complex formatting, such as LaTeX text, diagrams, or images, are excluded from the dataset.
3. Combine with Ground Truth Answers: The relevant questions are combined with their corresponding ground truth answers in a JSON format for organization and efficiency.
4. Data Preprocessing:
    a. Obtain Study Materials: The system obtains study materials such as NCERT textbooks, Oswal Revision notes, and Concepts of Physics.
    b. Parse and Segment: The study materials are parsed and segmented into smaller, manageable chunks to address the context-window limitations of language models.
    c. Convert to Plain Text: Various file formats are converted into standardized plain text to ensure consistency across documents.
    d. Create Vector Embeddings: The text chunks are transformed into vector representations using an embedding model, such as GoogleGenerativeAIEmbeddings.
    e. Store in Vector Database: The vector embeddings are stored in a vector database, enabling efficient and scalable search capabilities.
5. Receive User Query: The system receives the user query and encodes it into a vector representation.
6. Retrieve Relevant Documents: The system searches the vector database for similar chunks based on the encoded user query. The top K relevant chunks are retrieved.
7. Combine with Query: The retrieved chunks are combined with the user query to form an enriched prompt.
8. Generate Response Using LLM:

a. The system generates a response using a large language model (LLM) such as Google Gemini-pro or GPT-3.5-turbo-0125.

b. The response is synthesized using both the prompt and the retrieved document to ensure accuracy and relevance.

9. Fallback Strategy:
    a. If the model fails to find an answer or responds with "Sorry, I don't have much information about it," a fallback strategy is implemented.
    b. The system switches to RAG-free mode, allowing the model to generate a response based solely on its internal knowledge base.
10. Display Response: The synthesized response is displayed to the user

3.3.3 Flowchart of System

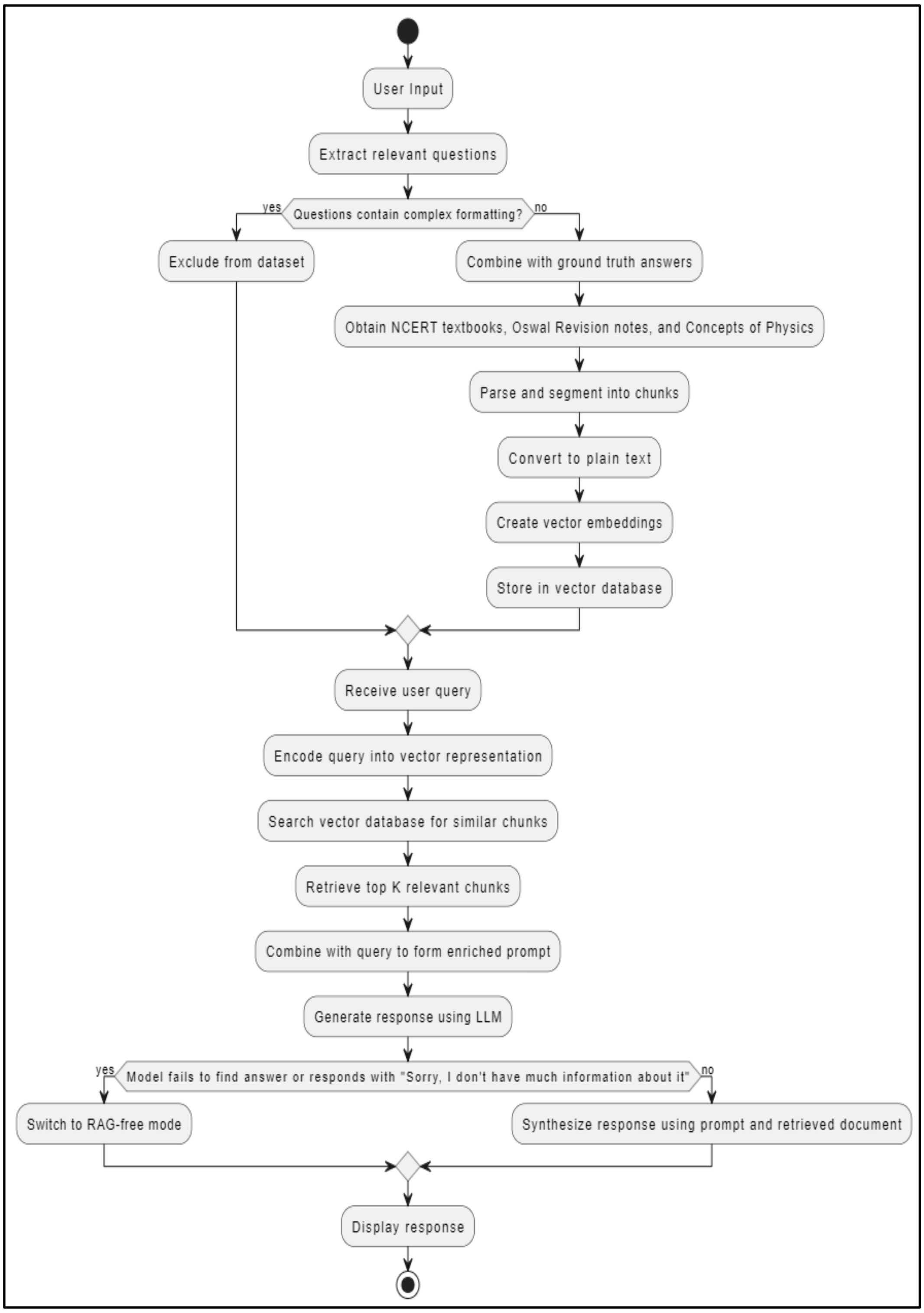

User Input
Extract relevant questions
Questions contain complex formatting?
yes
no
Exclude from dataset
Combine with ground truth answers
Obtain NCERT textbooks, Oswal Revision notes, and Concepts of Physics
Parse and segment into chunks
Convert to plain text
Create vector embeddings
Store in vector database
Receive user query
Encode query into vector representation
Search vector database for similar chunks
Retrieve top K relevant chunks
Combine with query to form enriched prompt
Generate response using LLM
Model fails to find answer or responds with "Sorry, I don't have much information about it"
yes
no
Switch to RAG-free mode
Synthesize response using prompt and retrieved document
Display response

Fig 3.3.3 : Flowchart of system

Chapter 4

System Requirements and Specification

4.1 Software requirements specifications:

4.1.1. Introduction:

The project is developed utilizing Python version 3.10, capitalizing on its extensive ecosystem of libraries and tools tailored for machine learning and natural language processing tasks. Python 3.10 offers a robust framework that facilitates efficient development and integration of complex algorithms and models required for the implementation of the proposed question-answering system. Leveraging Python's versatility and widespread adoption within the data science community ensures compatibility with various machine learning frameworks and libraries, thereby enhancing the project's flexibility and scalability.

The primary objective of the project is to construct a comprehensive question-answering system tailored specifically for the National Eligibility cum Entrance Test (NEET) examination. NEET serves as a pivotal evaluation for aspiring medical professionals, encompassing subjects such as physics (PHY), chemistry (CHEM), and biology (BIO). The project aims to harness state-of-the-art language models (LLMs) and Application Programming Interfaces (APIs) to develop an intelligent system capable of accurately responding to queries pertaining to NEET syllabi. By integrating cutting-edge technologies and leveraging advanced machine learning techniques, the system endeavors to provide students with a valuable resource for enhancing their understanding and preparation for the NEET examination.

4.1.2. Operating Environment:

The development environment employed Google Colab, incorporating a

T4 GPU to augment processing capabilities, thereby facilitating efficient execution of training and inference tasks involving extensive datasets and intricate models. Google Colab's integration of high-performance computing resources, coupled with the T4 GPU, enhances computational efficiency, accelerating model training and inference processes. This environment fosters seamless development and experimentation with advanced machine learning algorithms and models, enabling the exploration of complex neural architectures and optimization techniques.

Operating within Google Colab's runtime environment ensures compatibility with Google's cloud services and resources, further augmenting the project's operational capabilities. The runtime environment provides a robust infrastructure for executing compute-intensive tasks, leveraging Google's scalable cloud infrastructure to accommodate diverse computing requirements. Additionally, integration with Google's cloud services facilitates streamlined data management, collaboration, and deployment processes, enhancing the overall efficiency and productivity of the project. By harnessing the capabilities of Google Colab's runtime environment, the project maximizes resource utilization and scalability, thereby optimizing the development and deployment of the question-answering system for the NEET examination.

4.1.3. Functional Requirements:

1. Google Gemini-Pro API:

Utilized for advanced natural language understanding and question-answering capabilities, the Google Gemini-Pro API serves as a foundational component of the question-answering system. This API leverages cutting-edge natural language processing techniques to analyze and interpret user queries, enabling the system to generate accurate and contextually relevant responses. By harnessing Google's advanced AI capabilities, including semantic understanding and entity recognition, the Gemini-Pro API enhances the system's ability to comprehend complex language patterns and extract pertinent information from diverse textual inputs.

2. OpenAI GPT-3.5-Turbo-0125 API:

Integrated for its powerful language generation and contextual understanding features, the OpenAI GPT-3.5-Turbo-0125 API enriches the question-answering system with state-of-the-art natural language processing capabilities. This API empowers the system to generate coherent and contextually appropriate responses to user queries, drawing upon a vast repository of linguistic knowledge and contextual understanding. By leveraging OpenAI's sophisticated language models, the GPT-3.5-Turbo-0125 API enhances the system's ability to provide informative and relevant answers across a wide range of topics and domains.

3. Document Collection:

The question-answering system relies on a comprehensive collection of NEET textbooks from various subjects, including Biology, Chemistry, and Physics. Additionally, supplementary resources such as Oswal revision notes for Physics and the HC Verma book are incorporated to augment the system's knowledge base. This extensive document collection serves as a foundational resource for the system, enabling it to access authoritative information and reference materials essential for generating accurate responses to user queries.

4. Libraries Used:

- LangChain: Utilized for language model chaining and prompt engineering, LangChain facilitates the integration of multiple language models to enhance the system's linguistic capabilities.
- Google GenAI: Employed for Google's generative AI capabilities, Google GenAI enables the system to generate diverse and contextually relevant responses to user queries.
- Pillow: Serving as an image processing library, Pillow enables the system to handle image data seamlessly, facilitating the incorporation of visual information into the question-answering process.
- PyPDF2 and PyPDF: Used for PDF document parsing and manipulation, PyPDF2 and PyPDF enable the system to extract textual information from PDF documents, enriching its knowledge base with diverse textual resources.
- Pydantic: Utilized for data validation and modeling, Pydantic ensures the integrity and consistency of data inputs and outputs within the system.
- OpenAI: Serving as a library for interacting with OpenAI's language models, the

OpenAI library provides essential functionalities for accessing and utilizing the capabilities of OpenAI's language models.

- ChromaDB: Employed for creating and managing subject-specific vector databases, ChromaDB facilitates efficient storage and retrieval of information relevant to specific subjects within the NEET syllabus.
- TikToken: Serving as a tokenization library for text processing, TikToken enables the system to tokenize textual inputs, facilitating efficient text processing and analysis.
- OpenCV-Python: Used for image processing tasks, OpenCV-Python enables the system to perform various image processing operations, enhancing its ability to interpret and analyze visual information.
- Other standard libraries:Including os, base64, uuid, json, etc., these standard libraries provide essential functionalities for file manipulation, data encoding/decoding, unique identifier generation, and JSON data processing within the system.

4.1.4. Other Non-functional Requirements:

1. Performance:

The system was meticulously designed with a focus on achieving high performance and scalability. Leveraging GPU acceleration and employing optimized code structures, the system ensures efficient processing of large datasets and complex queries. By harnessing the computational power of GPUs, the system accelerates key tasks such as model training, inference, and data processing, thereby enhancing overall performance and responsiveness. Additionally, the system incorporates caching mechanisms and parallel processing techniques to further optimize resource utilization and minimize latency, ensuring seamless user interactions and timely responses to queries.

2. Security:

Robust security measures have been implemented to safeguard sensitive data, API keys, and user interactions within the question-answering system.

Encryption protocols are employed to protect data both in transit and at rest, mitigating the risk of unauthorized access or data breaches. Access controls and authentication mechanisms are enforced to restrict system access to authorized users only, preventing unauthorized access to sensitive resources. Furthermore, regular security audits and vulnerability assessments are conducted to identify and address potential security vulnerabilities proactively. These measures collectively ensure the confidentiality, integrity, and availability of the system, instilling trust and confidence among users regarding the security of their data and interactions.

3. **Reliability:**

The question-answering system undergoes extensive testing and validation procedures to ensure its reliability and accuracy, particularly in handling diverse subject domains and user queries. Rigorous testing methodologies, including unit testing, integration testing, and regression testing, are employed to verify the correctness and robustness of system functionalities across various scenarios and edge cases. Additionally, the system is subjected to performance testing under simulated load conditions to assess its scalability and responsiveness under peak usage scenarios. Continuous monitoring and error logging mechanisms are implemented to detect and address potential issues proactively, minimizing downtime and ensuring uninterrupted service availability. These reliability measures ensure that the question-answering system consistently delivers accurate and dependable responses, meeting the expectations and requirements of its users.

4.1.5. Product Perspective:

The question-answering system operates within the educational and exam preparation domains, leveraging Retrieval-Augmented Generation (RAG) techniques to specialize in competitive exams in India. Its primary perspective is to emulate human intelligence in question-answering domains, particularly focusing on NEET and similar exams.

It interfaces with external APIs, libraries, and data sources to enhance functionality, accuracy, and coverage of relevant information, ensuring comprehensive insights and accurate answers to NEET exam-related queries.

4.1.6. Product Function:

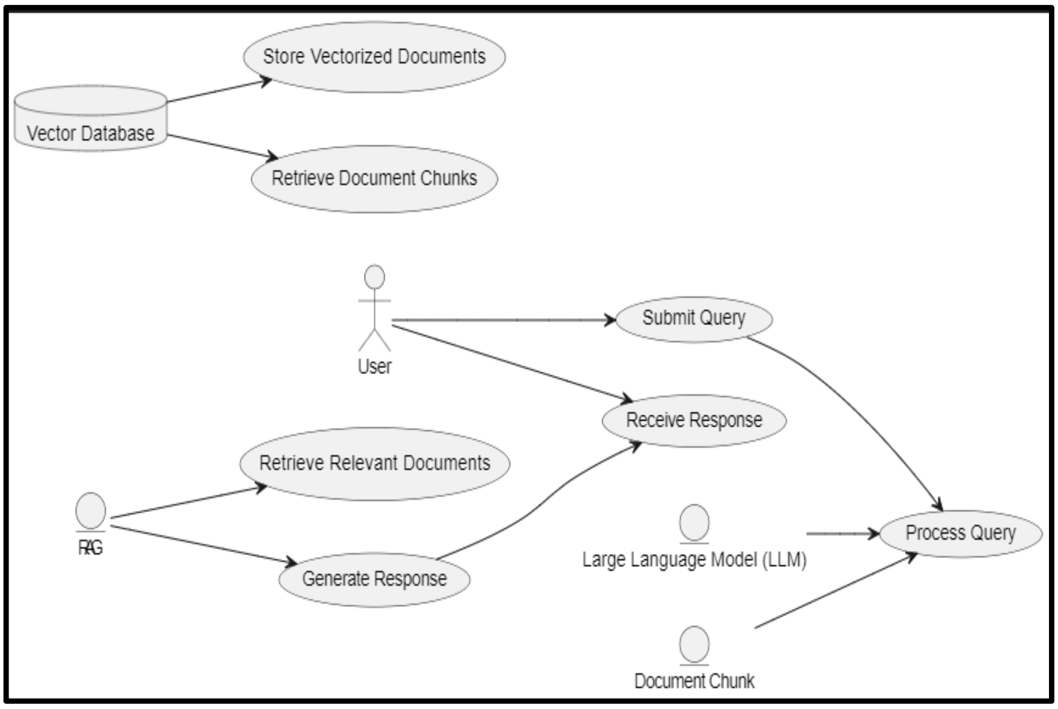
The system operates as a comprehensive answering tool tailored for competitive exams like NEET, integrating advanced language models, API integrations, and data processing capabilities. It facilitates seamless user interactions and provides valuable educational insights. Moreover, its adaptability allows for easy transition to other exams, achieved by modifying the vector data corpus and question sets to suit different benchmarks or exam formats efficiently.

This flexibility ensures the system's relevance and applicability across various educational domains and exam contexts, catering to diverse user needs and requirements. This structured report highlights the key components, technologies, and functionalities of the NEET question-answering system, providing a comprehensive overview of its development environment, operational requirements, and functional capabilities.

Chapter 5

Project Analysis and Design

5.1 Use Case Diagram



Use case diagram:

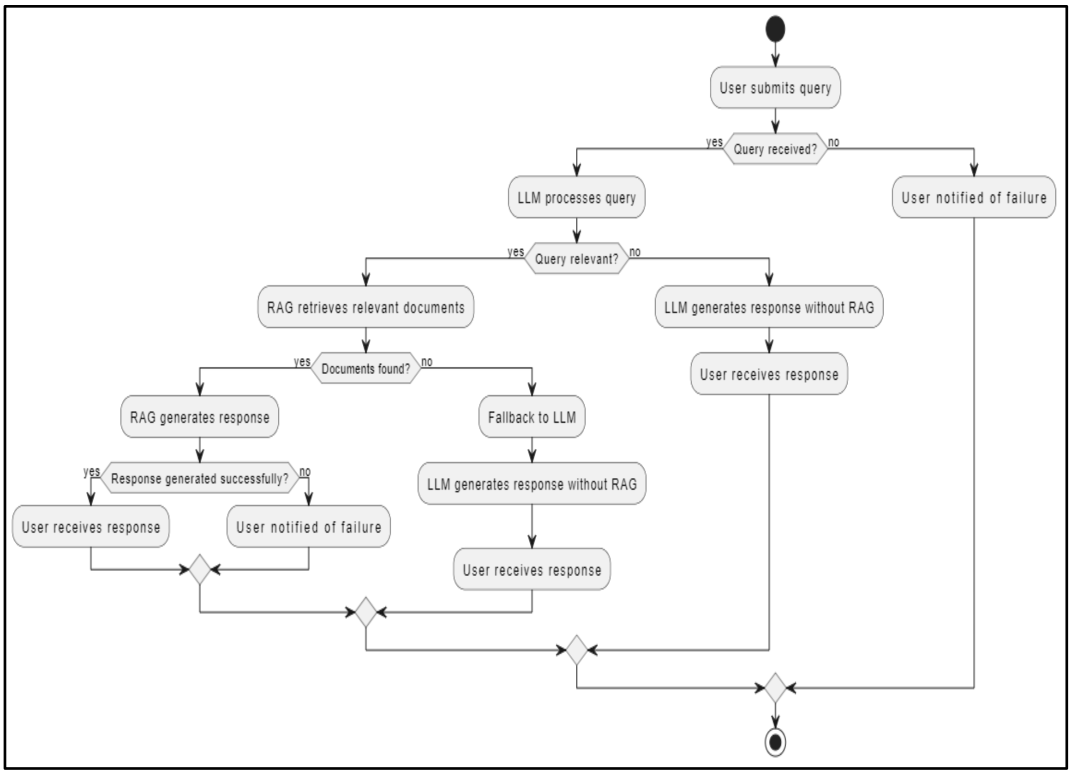
1. User: Represents the person interacting with the system by submitting queries and receiving responses.
2. LLM (Large Language Model): Refers to the LLM component of the system responsible for processing the user's query.
3. RAG (Retrieval-Augmented Generation): Represents the RAG component of the system, which retrieves relevant documents and generates responses based on the user's query.
4. Vector Database: Represents the database where vectorized documents are stored. This component facilitates efficient retrieval of document chunks during the RAG process.

5. Document Chunk: Represents the individual chunks of documents retrieved from the vector database. These document chunks are processed to extract relevant information during the response generation phase.

Interactions:

- Submit Query: The user submits a query to the system for processing.
- Process Query: Both the LLM and Document Chunk components process the user's query to extract relevant information.
- Retrieve Relevant Documents: The RAG component retrieves relevant documents from the vector database based on the user's query.
- Generate Response: The RAG component generates a response based on the retrieved documents and the user's query.
- Receive Response: The user receives the response generated by the system

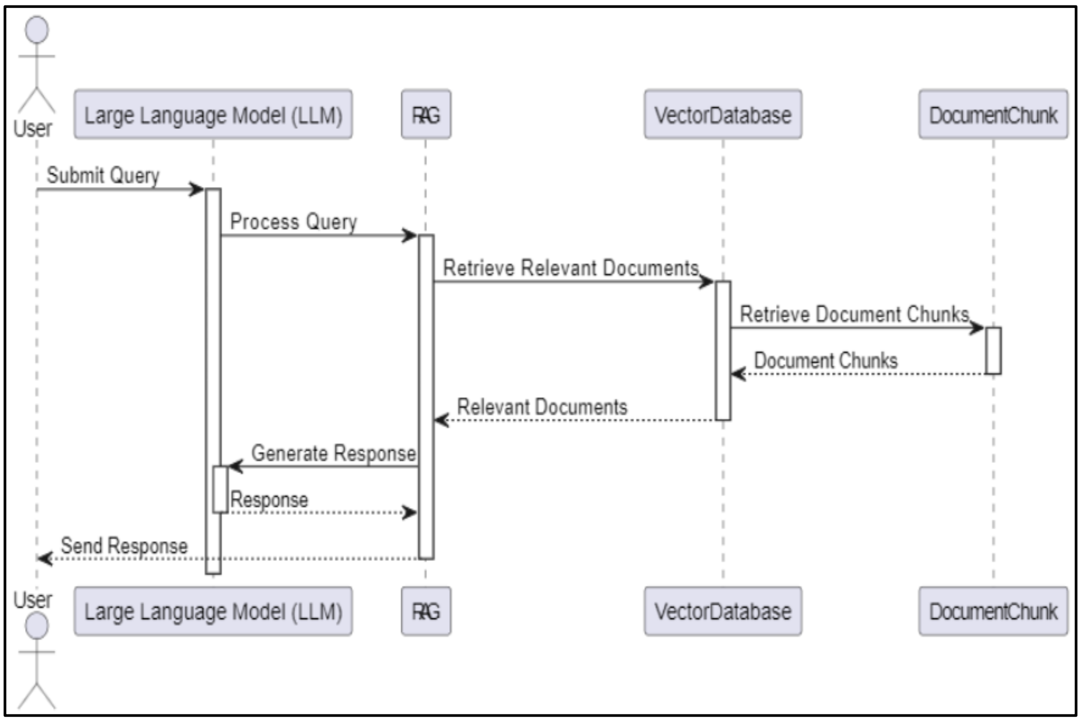
5.2 Activity Diagram



1. User submits query: This is the starting point of the system workflow. The user submits a query to the system, initiating the question-answering process.
2. Query received?: The system checks if the query has been successfully received.

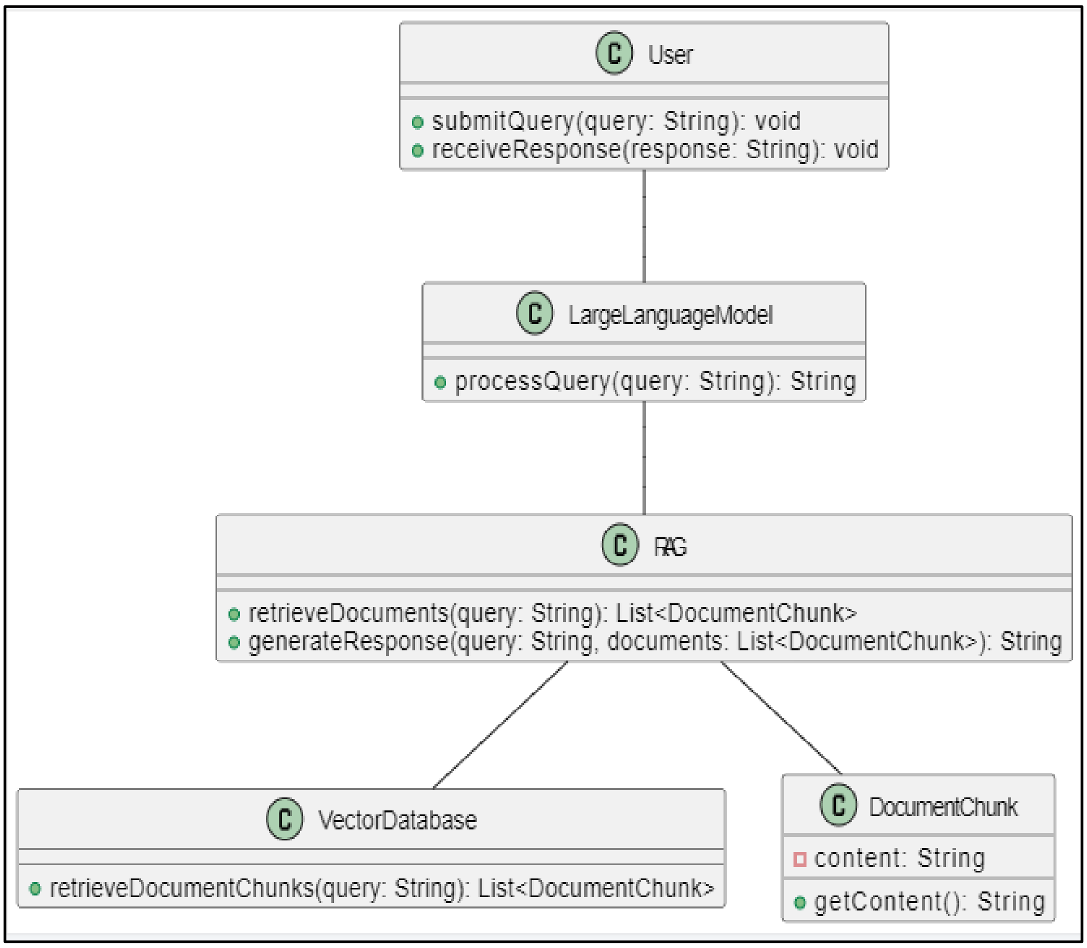
3. LLM processes query: If the query has been received, it is passed to the Large Language Model (LLM) for processing. The LLM analyzes the query to understand its content and context.
4. Query relevant?: After processing the query, the system determines if it is relevant to the task at hand. If the query is deemed relevant, the system proceeds with the Retrieval-Augmented Generation (RAG) process.
5. RAG retrieves relevant documents: In this step, the RAG component of the system retrieves relevant documents from the database based on the processed query. These documents serve as the basis for generating the response.
6. Documents found?: The system checks if relevant documents have been successfully retrieved based on the query. If documents are found, the system proceeds with response generation using RAG.
7. RAG generates response: Using the retrieved documents and the processed query, RAG generates a response that is contextually relevant and accurate. This response is crafted based on the information contained in the documents.
8. Response generated successfully?: After generating the response, the system checks if it was successful. If the response is successfully generated, it is sent to the user.
9. User receives response: The user receives the response generated by the system, completing the question-answering process.
10. Fallback to LLM: If the query is determined to be irrelevant or if no relevant documents are found during the RAG process, the system falls back to using only the LLM to generate a response. This ensures that the user receives a response even in cases where RAG is not applicable or unsuccessful.
11. User notified of failure: If the system encounters any failures during the process, such as the inability to process the query or retrieve relevant documents, the user is notified of the failure, and no response is provided.

5.3.1 Sequence diagram



1. User submits query: The user initiates the process by submitting a query to the system.
2. LLM processes query: The Large Language Model (LLM) receives the query from the user and processes it to understand its context.
3. RAG retrieves relevant documents: The RAG component of the system retrieves relevant documents from the vector database based on the processed query.
4. Document chunks retrieved: The DocumentChunk module retrieves the necessary document chunks from the vector database.
5. Relevant documents provided to RAG: The retrieved document chunks are provided to the RAG component for further processing.
6. RAG generates response: Using the retrieved documents and the processed query, RAG generates a response.
7. Response sent to user: The response generated by RAG is sent back to the user.

5.3.2 Class diagram

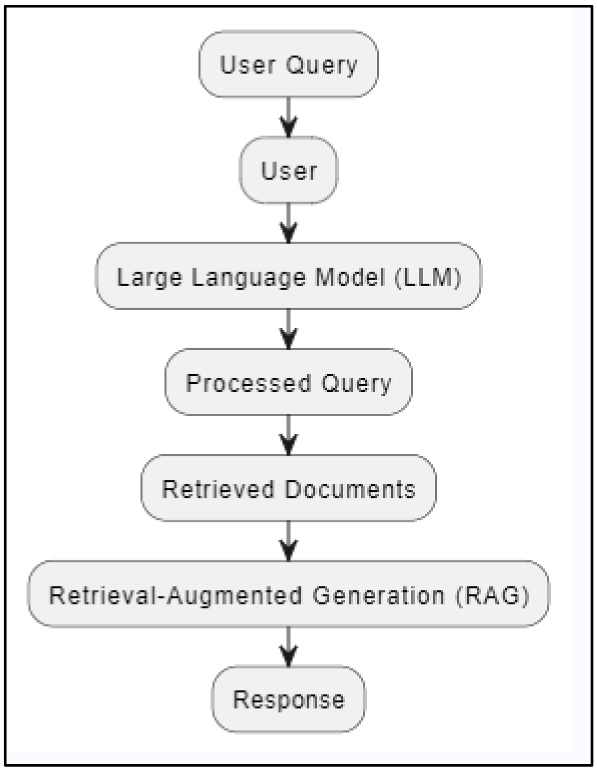

C User
submitQuery(query: String): void
receiveResponse(response: String): void
C LargeLanguageModel
processQuery(query: String): String
C RAG
retrieveDocuments(query: String): List<DocumentChunk>
generateResponse(query: String, documents: List<DocumentChunk>): String
C VectorDatabase
retrieveDocumentChunks(query: String): List<DocumentChunk>
C DocumentChunk
content: String
getContent(): String

1. User: Represents the user interacting with the system.
   a. submitQuery(query: String): Method to submit a query to the system.
   b. receiveResponse(response: String): Method to receive a response from the system.
2. LargeLanguageModel:
   a. Represents the large language model component of the system.
   b. processQuery(query: String): Method to process a query and return the processed query.
3. RAG (Retrieval-Augmented Generation): Represents the retrieval-augmented generation component of the system.
   a. retrieveDocuments(query: String): Method to retrieve relevant documents based on a query.

   b. generateResponse(query: String, documents: List<DocumentChunk>): Method to generate a response based on a query and a list of document chunks.

4. VectorDatabase: Represents the vector database component of the system.
   a. retrieveDocumentChunks(query: String): Method to retrieve document chunks based on a query.
5. DocumentChunk: Represents a chunk of a document retrieved from the database.
   a. getContent(): Method to get the content of the document chunk

5.4 Data flow diagram:


User Query
User
Large Language Model (LLM)
Processed Query
Retrieved Documents
Retrieval-Augmented Generation (RAG)
Response

1. User Query: Represents the query submitted by the user to the system.
2. User: Represents the user interacting with the system.
3. Large Language Model (LLM): Represents the component responsible for processing the user query.
4. Processed Query: Represents the query processed by the Large Language Model.

5. Retrieved Documents: Represents the documents retrieved by the system based on the processed query.
6. Retrieval-Augmented Generation (RAG): Represents the component responsible for generating a response based on the retrieved documents.
7. Response: Represents the response generated by the system and provided to the user

5.5 Team Organization

5.5.1. Team Structure

Our team:- Our team consists of Researchers/Developers, an internal guide, external guide.

Researchers:

1. Dhairya Thakkar
2. Dhiren Oswal
3. Soham Tolwala
4. Pushpak Suryawanshi

Internal Guide

1. Mr. Swapnil Shinde
2. Dr. Nakul Sharma (Panel Member)
3. Mrs. Gitanjali Yadav (Panel Member)

5.5.2. Task Summary

The project task set for the NEET-QA involves a series of activities essential for the successful development, implementation of the system. Here is an overview of key tasks:

1. Requirements Gathering: Conduct literature surveys to identify and document project requirements. Conduct student interviews to understand nuances.
2. Data Gathering: Select and filter books to be vectorised, and questions from NEET paper to be used as queries.

3. Vector Database Creation: Pre process the books, chunk the content, vectorize it and store it as a Vector store
4. LLM Interfacing: Connect the APIs, and initialize the model with prompt templates
5. Initiate a retriever: Instantiate the vector store in retriever mode to fetch relevant content based on the query.
6. Connect Retriever to LLM: Pass the retrieved chunks to the retriever.
7. Make the query bank: Parse the NEET previous year's paper and filter the feasible questions. combine the answers.
8. Answer generation:
   a. Iteratively feed query bank to the embedding model and send the embedded query to the vector store
   b. Retrieve content from the vector store
   c. Query and fetched document is parsed to the LLM template- Perform integration testing to ensure seamless interactions between functionalities.
9. Testing: Check the answer generated against the ground truth available in the Previous year's question paper
10. Evaluate Score- Define a standard metric and evaluate the performance of the system on the test
11. Documentation: Create comprehensive project documentation, including manuals for reproducibility, design specifications, and evaluation framework.

5.5.3. Tools

1. Google Meet: To discuss methodologies and discuss the result
2. Gmail: Maintain official Communication
3. Jupyter Notebook: Maintain Code and documentation
4. Git - Pushing the the actual implementation (code)

Chapter 6
Implementation Software Testing

6.1 Implementation

6.1.1 - Document ingestion and creating vector store

Firstly, we’ll demonstrate the workflow of our implementation for one particular subject i.e. PHYSICS as we have followed the same approach for the remaining two subjects i.e. CHEMISTRY and BIOLOGY.

The code block initiates the document ingestion process by specifying the directory name as 'PHY'. It checks if the directory already exists in the Colab session storage. If the directory does not exist, it creates the directory 'PHY' and prints a success message confirming the creation. If the directory already exists, it prints a message indicating that the directory 'PHY' is already present. This step ensures that the required directory is available for storing the PHY documents in the Colab session storage.

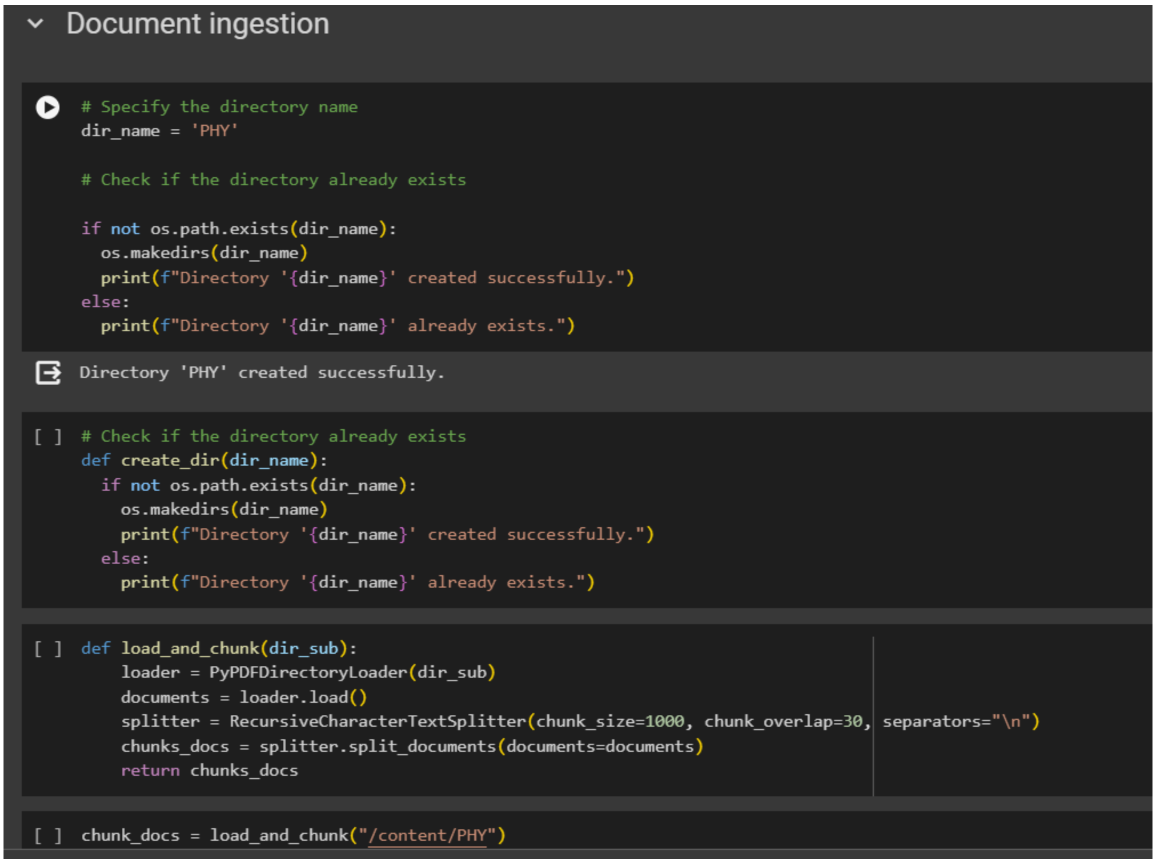
After creating the directory, you would typically upload documents related to physics or any other specified category to the Colab session storage. This step is not explicitly shown in the provided code block but would typically involve using the Colab interface to upload files or mounting Google Drive to access document files stored there.

The `load_and_chunk` function is designed to facilitate the loading and chunking of documents within a specified directory. It takes the directory name (`dir_sub`) as input, indicating the subdirectory where the documents are stored.

Here's a detailed explanation of the function:

1. PyPDFDirectoryLoader Initialization

The function begins by initializing a `PyPDFDirectoryLoader` object named `loader`. This loader is responsible for loading documents from the specified directory (`dir_sub`). It uses PyPDF2, a Python library for working with PDF files, to handle the loading process.


Document ingestion

```
# Specify the directory name
dir_name = 'PHY'

# Check if the directory already exists

if not os.path.exists(dir_name):
  os.makedirs(dir_name)
  print(f"Directory '{dir_name}' created successfully.")
else:
  print(f"Directory '{dir_name}' already exists.")
```

```
Directory 'PHY' created successfully.
```

```
# Check if the directory already exists
def create_dir(dir_name):
  if not os.path.exists(dir_name):
    os.makedirs(dir_name)
    print(f"Directory '{dir_name}' created successfully.")
  else:
    print(f"Directory '{dir_name}' already exists.")
```

```
def load_and_chunk(dir_sub):
    loader = PyPDFDirectoryLoader(dir_sub)
    documents = loader.load()
    splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=30, separators="\n")
    chunks_docs = splitter.split_documents(documents=documents)
    return chunks_docs
```

```
chunk_docs = load_and_chunk("/content/PHY")
```

2.   Document Loading

The `load()` method is then called on the `loader` object to load all the documents present in the specified directory. These documents could be in PDF format, as indicated by the use of PyPDF2.

3.   Recursive Chunking

After loading the documents, the function proceeds to chunk them using a `RecursiveCharacterTextSplitter` object named `splitter`. This chunking process involves breaking down each document into smaller, manageable chunks to facilitate processing and analysis.

4.   Chunk Parameters

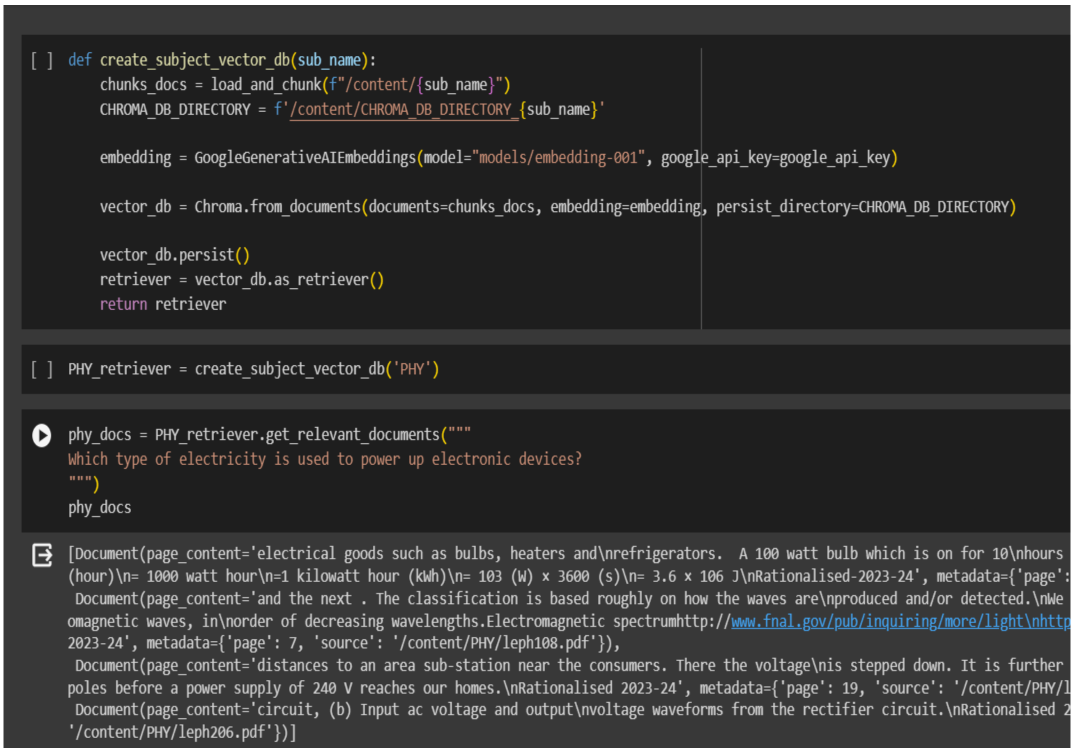
The `chunk_size` parameter specifies the maximum size of each chunk, ensuring that chunks are not too large, which could impact processing efficiency. The `chunk_overlap` parameter defines the overlap between consecutive chunks, providing continuity in the text analysis process. Additionally, the `separators` parameter specifies the characters used as separators between chunks, often newline characters (`\n`) in text-based documents.

5. Chunked Documents Return

Finally, the function returns the chunked documents (`chunks_docs`) obtained after the chunking process. These chunked documents are now in a format suitable for further analysis, such as text processing, natural language understanding, or other data-driven tasks.

In summary, the `load_and_chunk` function automates the process of loading documents from a specified directory, chunking them into manageable segments, and preparing them for subsequent analysis or processing, contributing to efficient document handling within a data processing pipeline.

Next we execute the `load_and_chunk` function with the argument "/content/PHY", which specifies the directory path where the PHY documents are located. This line of code initiates the process of loading and chunking the documents within the specified directory using the defined function. The resulting `chunk_docs` variable holds the chunked documents, which are now ready for further analysis or processing. This step is crucial in preparing the documents for tasks such as text mining, information retrieval, or natural language processing, enhancing the efficiency of data handling and analysis workflows.



```
[ ] def create_subject_vector_db(sub_name):
        chunks_docs = load_and_chunk(f"/content/{sub_name}")
        CHROMA_DB_DIRECTORY = f'/content/CHROMA_DB_DIRECTORY_{sub_name}'

        embedding = GoogleGenerativeAIEmbeddings(model="models/embedding-001", google_api_key=google_api_key)

        vector_db = Chroma.from_documents(documents=chunks_docs, embedding=embedding, persist_directory=CHROMA_DB_DIRECTORY)

        vector_db.persist()
        retriever = vector_db.as_retriever()
        return retriever
```

```
[ ] PHY_retriever = create_subject_vector_db('PHY')
```

```
phy_docs = PHY_retriever.get_relevant_documents("""
Which type of electricity is used to power up electronic devices?
""")
phy_docs
```

```
[Document(page_content='electrical goods such as bulbs, heaters and\nrefrigerators.  A 100 watt bulb which is on for 10\nhours
(hour)\n= 1000 watt hour\n=1 kilowatt hour (kWh)\n= 103 (W) x 3600 (s)\n= 3.6 x 106 J\nRationalised-2023-24', metadata={'page':
 Document(page_content='and the next . The classification is based roughly on how the waves are\nproduced and/or detected.\nWe
omagnetic waves, in\norder of decreasing wavelengths.Electromagnetic spectrumhttp://www.fnal.gov/pub/inquiring/more/light\nhttp
2023-24', metadata={'page': 7, 'source': '/content/PHY/leph108.pdf'}),
 Document(page_content='distances to an area sub-station near the consumers. There the voltage\nis stepped down. It is further
poles before a power supply of 240 V reaches our homes.\nRationalised 2023-24', metadata={'page': 19, 'source': '/content/PHY/l
 Document(page_content='circuit, (b) Input ac voltage and output\nvoltage waveforms from the rectifier circuit.\nRationalised 2
'/content/PHY/leph206.pdf'})]
```

The `create_subject_vector_db` function is designed to create a subject-specific vector database using the documents from a specified directory, incorporating advanced embedding techniques and persistent storage mechanisms. Here's a detailed explanation of each step within the function:

1. Loading and Chunking Documents

   The function first calls the `load_and_chunk` function, passing the directory path for the subject (`/content/{sub_name}`). This step loads the documents from the specified directory and chunks them into smaller segments, preparing them for vectorization and storage.

2. Setting Up Chroma Database Directory:

   The variable `CHROMA_DB_DIRECTORY` is initialized to store the directory path for the Chroma vector database specific to the subject. This directory will hold the vectorized representations of the chunked documents, facilitating efficient retrieval and analysis.

3. Embedding Initialization

   An instance of the GoogleGenerativeAIEmbeddings class is created, specifying the embedding model to be used (`model="models/embedding-001"`) and providing the Google API key (`google_api_key`) for authentication and access to

Google's embedding services. This embedding model is crucial for converting textual data into dense vector representations suitable for similarity comparisons and retrieval tasks.

4. Vector Database Creation

The Chroma vector database (`vector_db`) is initialized using the Chroma.from_documents method. This method takes the chunked documents (`chunks_docs`), the initialized embedding model (`embedding`), and the specified directory for persistent storage (`persist_directory=CHROMA_DB_DIRECTORY`) as input parameters. The vector database is created based on these inputs, containing vectorized representations of the documents in the subject-specific directory.

5. Persistence and Retriever Setup

The `vector_db.persist()` method is called to persistently store the vectorized documents in the Chroma vector database directory. This ensures that the vector representations remain accessible and usable across sessions or system restarts. Additionally, the vector database is converted into a retriever (`retriever`) using the `as_retriever()` method, enabling efficient retrieval operations based on similarity scores and queries.

6. Return Value

Finally, the function returns the initialized retriever (`retriever`), allowing for seamless retrieval and analysis of documents within the subject-specific vector database.

In summary, the `create_subject_vector_db` function orchestrates the creation of a subject-specific vector database using advanced embedding techniques and persistent storage mechanisms, providing a foundation for efficient document retrieval and analysis tailored to the specified subject area.

The code `PHY_retriever = create_subject_vector_db('PHY')` initializes a subject-specific vector database named 'PHY_retriever' for the subject 'PHY'. This database contains vectorized representations of documents related to physics, facilitating efficient retrieval operations based on similarity scores and user queries.

Next, the code `phy_docs = PHY_retriever.get_relevant_documents("""Which type of electricity is used to power up electronic devices?""")` demonstrates the retrieval of relevant documents from the 'PHY_retriever' database based on a user query. The query, "Which type of electricity is used to power up electronic devices?", is passed to the `get_relevant_documents` method of the 'PHY_retriever' object.

The `get_relevant_documents` method calculates the similarity between the user query vector and the vectorized documents in the database. It retrieves documents that are most relevant to the query based on their vector similarity scores. In this case, the query is related to electricity and electronic devices, so the method fetches documents from the physics domain that discuss the types of electricity used in powering electronic devices.

The variable `phy_docs` stores the retrieved relevant documents, allowing further analysis or processing of the information contained within these documents. This process showcases the functionality of the vector database and retrieval system, enabling users to obtain contextually relevant information based on their queries, enhancing the efficiency and accuracy of information retrieval tasks within the specified subject domain.

6.1.2 - Initializing LLM chains and prompt templates

Prompt templates crafted:

BIOLOGY prompt:

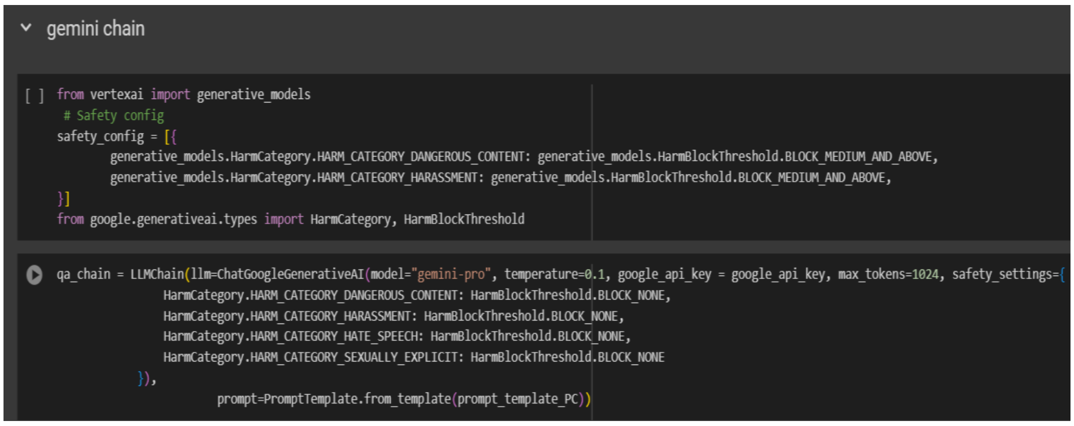
```
prompt_template_bio = """
You are a question answering bot which is specialized in BIOLOGY.
Answer the following Multiple Choice based question which has
one single correct answer from the four options given in the
question itself. you have to accurately choose the correct option
and return it.
Choose the answer according to the context provided below.

GOLDEN RULE: You will be evaluated by your predicted option and
hence focus on returning only the option.

Context: {context}
Question: {question}

Don't answer if the context provided to you is irrelevant, if
```

```
you are not sure
and confident and decline to answer and say "Sorry, I don't have
much information about it."
Include a brief reason for why you chose a particular option.
Answer:
"""
```

PHYSICS AND CHEMISTRY prompt:

```
prompt_template_PC = """
You are a question answering bot which is specialized in
subjects: Physics or Chemistry. Answer the following Multiple
Choice based question which has one single correct answer from
the four options given in the question itself. you have to
accurately choose the correct option and return it.

Chain of thought process you should follow:
1. Read and Understand the question provided and check if it's
numerical or a conceptual question.
2. If question is NUMERICAL.
  2a. Extract the given values from the question and relate it
to the given appropriate quantities.
  2b. Fetch the appropriate formula from the context if available
that is provided below.
  2c. Solve for the given question using the formula and values.
  2d. Return the appropriate option that matches to the solution.
3. If question is conceptual
  3a. Make use of the relevant retrieved context provided below.
  3b. Solve the question using the concept.
  3c. Return the approapriate option that matches to the
solution.
```

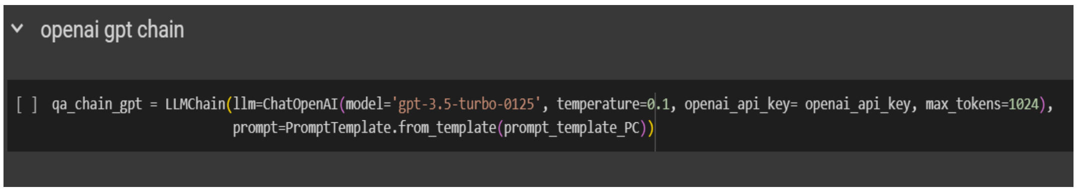
```
GOLDEN RULES
1. Always follow the chain of thought process.
2. You will be evaluated by your predicted option and hence focus
on returning only the option and nothing else.

Context: {context}
Question: {question}

NOTE: Don't answer if the context provided to you is irrelevant,
if you are not sure and confident then decline to answer and
return 'SORRY'."
      Include a brief reason for why you chose a particular
option.
"""
```



gemini chain

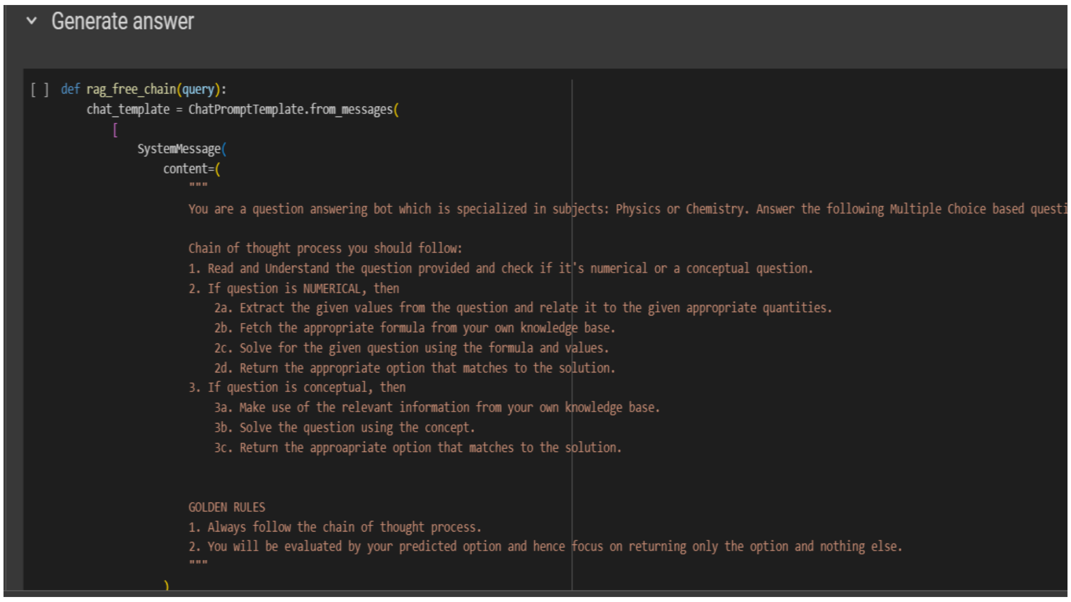
```
[ ] from vertexai import generative_models
     # Safety config
    safety_config = [{
            generative_models.HarmCategory.HARM_CATEGORY_DANGEROUS_CONTENT: generative_models.HarmBlockThreshold.BLOCK_MEDIUM_AND_ABOVE,
            generative_models.HarmCategory.HARM_CATEGORY_HARASSMENT: generative_models.HarmBlockThreshold.BLOCK_MEDIUM_AND_ABOVE,
    }]
    from google.generativeai.types import HarmCategory, HarmBlockThreshold
```

```
qa_chain = LLMChain(llm=ChatGoogleGenerativeAI(model="gemini-pro", temperature=0.1, google_api_key = google_api_key, max_tokens=1024, safety_settings={
                HarmCategory.HARM_CATEGORY_DANGEROUS_CONTENT: HarmBlockThreshold.BLOCK_NONE,
                HarmCategory.HARM_CATEGORY_HARASSMENT: HarmBlockThreshold.BLOCK_NONE,
                HarmCategory.HARM_CATEGORY_HATE_SPEECH: HarmBlockThreshold.BLOCK_NONE,
                HarmCategory.HARM_CATEGORY_SEXUALLY_EXPLICIT: HarmBlockThreshold.BLOCK_NONE
            }),
                    prompt=PromptTemplate.from_template(prompt_template_PC))
```



openai gpt chain

```
[ ] qa_chain_gpt = LLMChain(llm=ChatOpenAI(model='gpt-3.5-turbo-0125', temperature=0.1, openai_api_key= openai_api_key, max_tokens=1024),
                        prompt=PromptTemplate.from_template(prompt_template_PC))
```



```
[ ] def rag_free_chain(query):
        chat_template = ChatPromptTemplate.from_messages(
            [
                SystemMessage(
                    content=(
                        """
                        You are a question answering bot which is specialized in subjects: Physics or Chemistry. Answer the following Multiple Choice based questi

                        Chain of thought process you should follow:
                        1. Read and Understand the question provided and check if it's numerical or a conceptual question.
                        2. If question is NUMERICAL, then
                            2a. Extract the given values from the question and relate it to the given appropriate quantities.
                            2b. Fetch the appropriate formula from your own knowledge base.
                            2c. Solve for the given question using the formula and values.
                            2d. Return the appropriate option that matches to the solution.
                        3. If question is conceptual, then
                            3a. Make use of the relevant information from your own knowledge base.
                            3b. Solve the question using the concept.
                            3c. Return the approapriate option that matches to the solution.


                        GOLDEN RULES
                        1. Always follow the chain of thought process.
                        2. You will be evaluated by your predicted option and hence focus on returning only the option and nothing else.
                        """
                    )
```

The provided code segment initializes three different language model (LLM) chains and prompt templates for question-answering tasks in specialized domains. Let's break down each part and its functionality:

1. Safety Configuration:

   The safety_config variable specifies the safety settings for the LLM chains, categorizing harmful content and setting block thresholds accordingly. This ensures that generated responses adhere to safety standards, blocking harmful or inappropriate content.

2. LLM Chains Initialization:
   - qa_chain (Gemini-Pro):

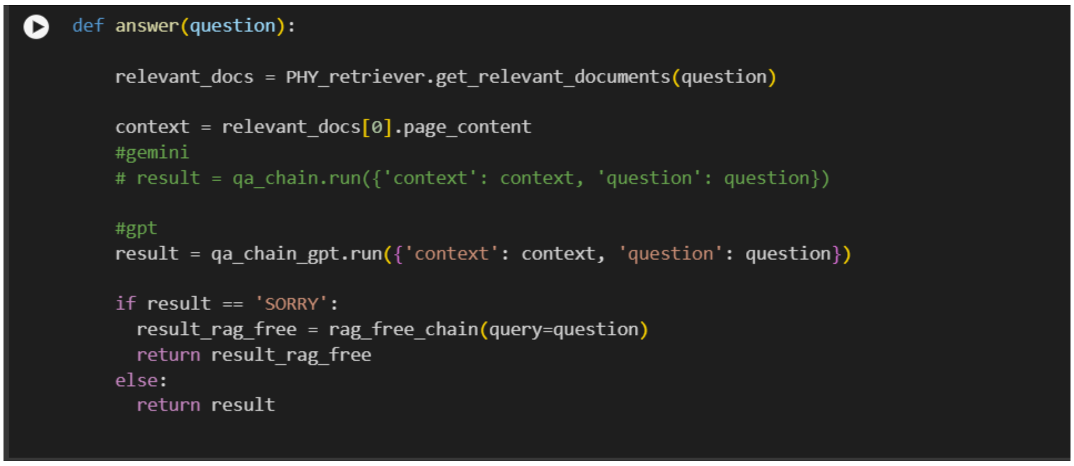
     This chain utilizes the ChatGoogleGenerativeAI model with the "gemini-pro" variant. Parameters such as temperature (0.1), Google API key (google_api_key), max_tokens (1024), and safety_settings are configured. The safety_settings block harmful content and harassment while allowing harmless content. The prompt_template_PC is used as a template for generating responses related to Physics or Chemistry questions.
   - qa_chain_gpt (GPT-3.5-Turbo):

     This chain uses the ChatOpenAI model with the "gpt-3.5-turbo-0125" variant. Parameters like temperature (0.1), OpenAI API key (openai_api_key), and max_tokens (1024) are set. This chain is designed for general question-answering tasks.

- rag_free_chain (RAG-Free):
  This function defines a free-form prompt template for question-answering. It guides the LLM in understanding the question type (numerical or conceptual) and provides a chain of thought process for answering questions accurately. The template includes rules and guidelines for the LLM's response format and behavior, emphasizing returning only the predicted option for MCQ-based questions.

3. Execution Flow (rag_free_chain):
   - The rag_free_chain function constructs a ChatPromptTemplate for handling user queries. It defines the structure and guidelines for the LLM's response, including a chain of thought process and golden rules for evaluation.
   - The chat_template guides the LLM through a structured approach for answering MCQ-based questions in subjects like Physics or Chemistry.
   - The llm variable initializes the ChatOpenAI model ('gpt-3.5-turbo-0125') for generating responses based on the chat_template.
   - The response generated by the llm is printed and returned, adhering to the guidelines and rules specified in the chat_template.

Overall, these LLM chains and prompt templates provide structured frameworks for question-answering tasks, ensuring safety, accuracy, and adherence to guidelines in generating responses across different specialized domains.



```
def answer(question):

    relevant_docs = PHY_retriever.get_relevant_documents(question)

    context = relevant_docs[0].page_content
    #gemini
    # result = qa_chain.run({'context': context, 'question': question})

    #gpt
    result = qa_chain_gpt.run({'context': context, 'question': question})

    if result == 'SORRY':
      result_rag_free = rag_free_chain(query=question)
      return result_rag_free
    else:
      return result
```

The answer function is designed to process user questions and generate responses using the initialized LLM chains and retriever. Here's a detailed explanation of how it works:

1. Get Relevant Documents:

   The function starts by retrieving relevant documents related to the user's question using the PHY_retriever.get_relevant_documents(question) method. This step fetches documents from the physics domain that are contextually relevant to the user's query.

2. Context Extraction:

   The context variable extracts the page content from the first relevant document retrieved. This context serves as the background information or knowledge base for generating the response.

3. LLM Chain Execution:

a. The function then runs the qa_chain_gpt (GPT-3.5-Turbo) chain using the run method. It passes the extracted context and the user's question as input to the LLM chain for processing.

b. If the result returned by qa_chain_gpt is 'SORRY', indicating that the LLM did not find a suitable answer, the function switches to the rag_free_chain for answering the query.

4. RAG-Free Chain Usage:

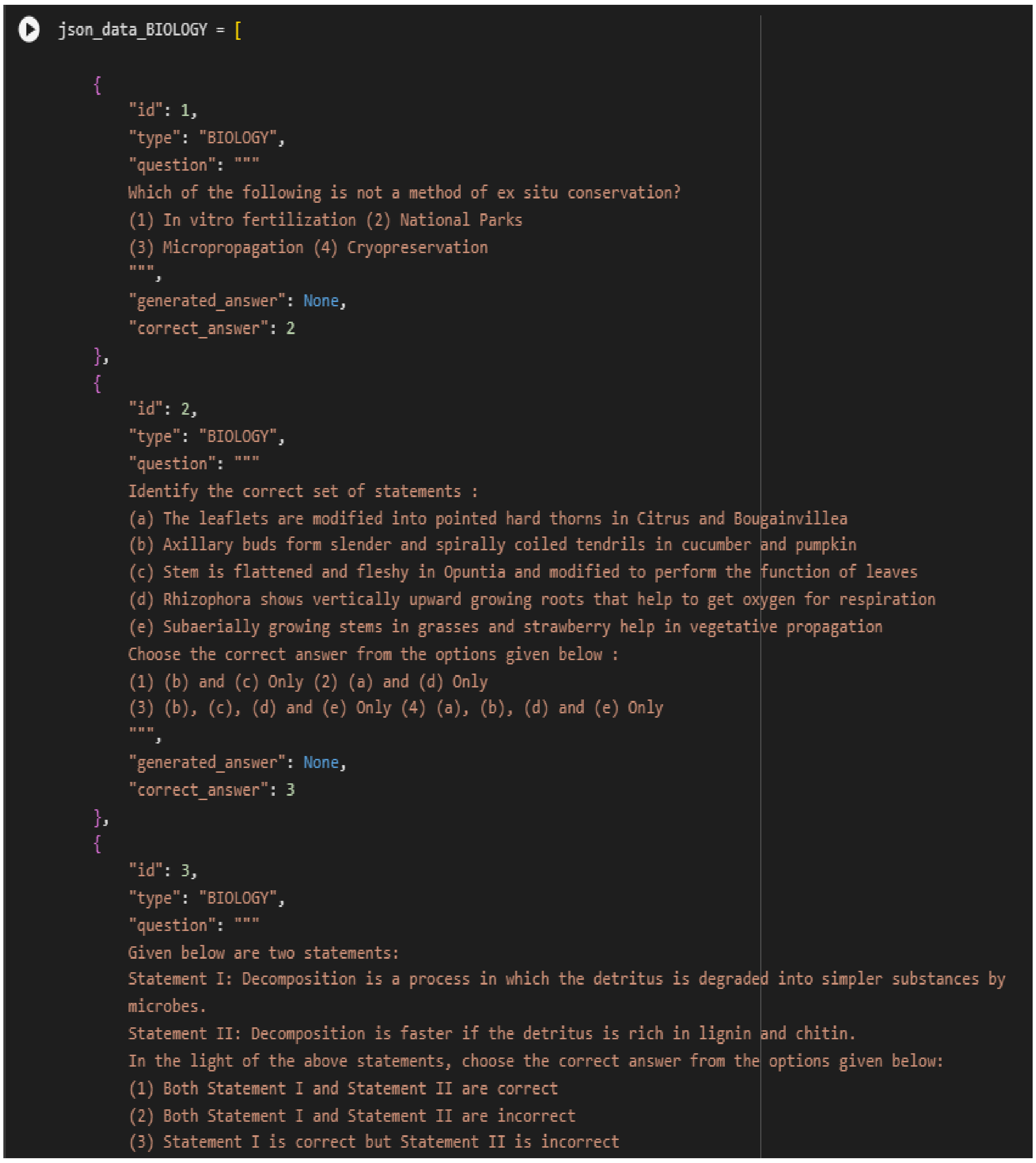
   If the LLM chain result is 'SORRY', the function calls the rag_free_chain(query=question) function. This function uses a free-form prompt template and the ChatOpenAI model ('gpt-3.5-turbo-0125') to generate a response based on the user's question and the specified chain of thought process.

5. Final Response Handling:
   The function returns the result generated by either qa_chain_gpt or rag_free_chain, depending on the availability of an answer from the LLM chain. If the LLM chain provides a suitable response, it is returned. Otherwise, the response from the rag_free_chain is returned to ensure a meaningful answer to the user's query.

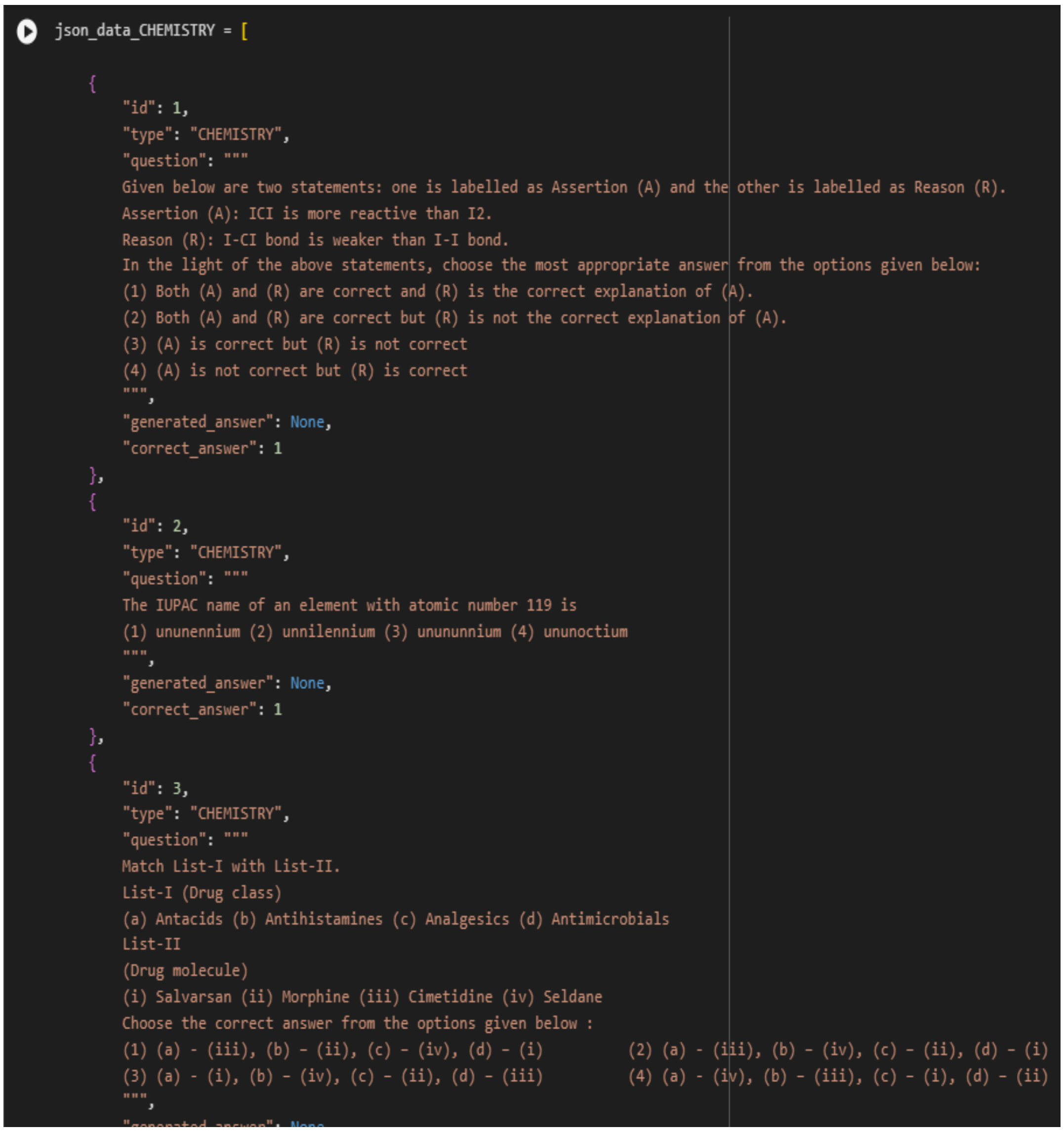
Overall, the answer function integrates the retrieval of relevant documents, context extraction, LLM chain execution, and fallback to a RAG-Free chain to ensure accurate and informative responses to user questions in the physics domain.

6.1.3 - Testing

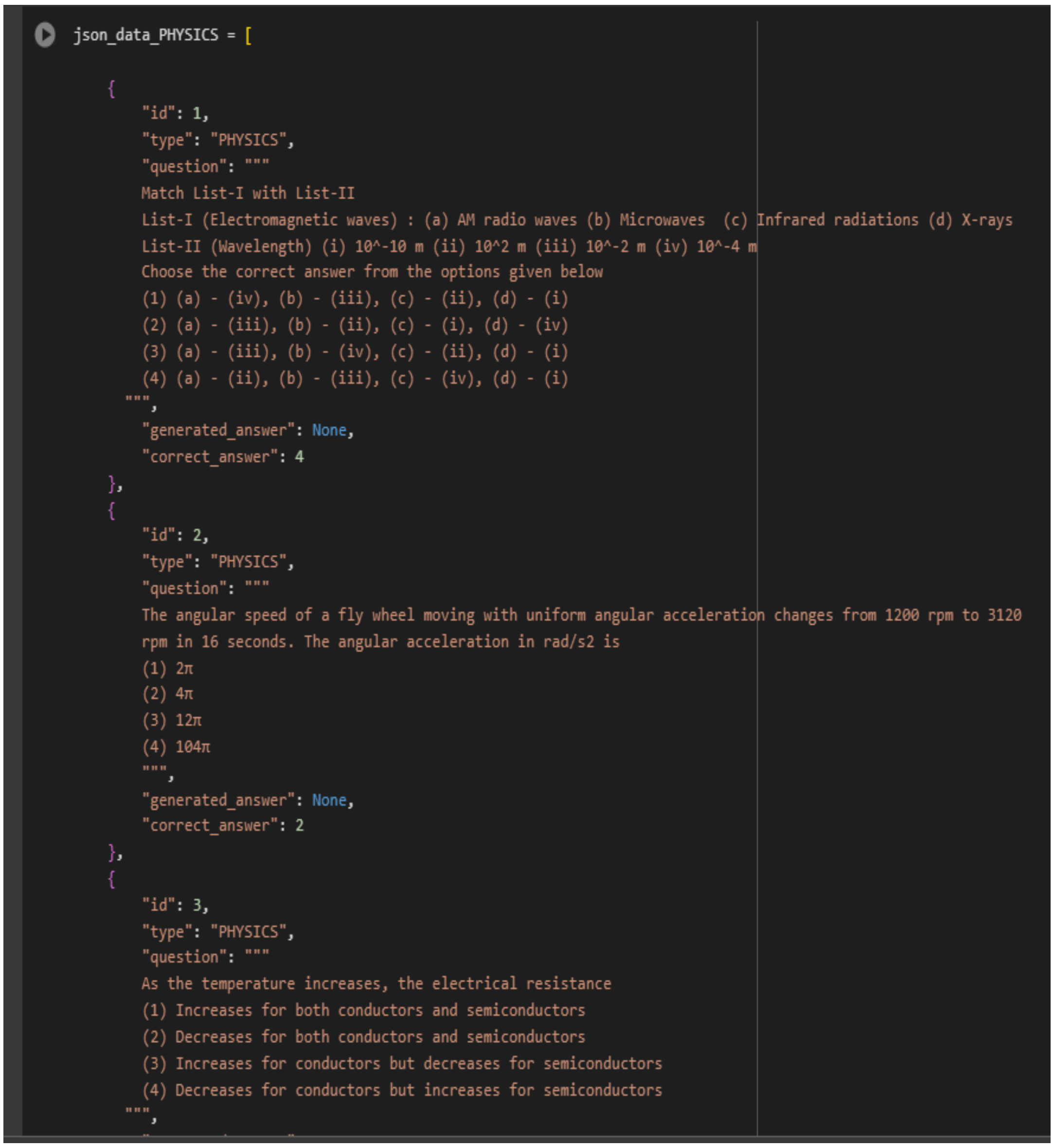
Now we test the answer function by passing a NEET multiple-choice questions:



```
json_data_BIOLOGY = [

        {
            "id": 1,
            "type": "BIOLOGY",
            "question": """
            Which of the following is not a method of ex situ conservation?
            (1) In vitro fertilization (2) National Parks
            (3) Micropropagation (4) Cryopreservation
            """,
            "generated_answer": None,
            "correct_answer": 2
        },
        {
            "id": 2,
            "type": "BIOLOGY",
            "question": """
            Identify the correct set of statements :
            (a) The leaflets are modified into pointed hard thorns in Citrus and Bougainvillea
            (b) Axillary buds form slender and spirally coiled tendrils in cucumber and pumpkin
            (c) Stem is flattened and fleshy in Opuntia and modified to perform the function of leaves
            (d) Rhizophora shows vertically upward growing roots that help to get oxygen for respiration
            (e) Subaerially growing stems in grasses and strawberry help in vegetative propagation
            Choose the correct answer from the options given below :
            (1) (b) and (c) Only (2) (a) and (d) Only
            (3) (b), (c), (d) and (e) Only (4) (a), (b), (d) and (e) Only
            """,
            "generated_answer": None,
            "correct_answer": 3
        },
        {
            "id": 3,
            "type": "BIOLOGY",
            "question": """
            Given below are two statements:
            Statement I: Decomposition is a process in which the detritus is degraded into simpler substances by
            microbes.
            Statement II: Decomposition is faster if the detritus is rich in lignin and chitin.
            In the light of the above statements, choose the correct answer from the options given below:
            (1) Both Statement I and Statement II are correct
            (2) Both Statement I and Statement II are incorrect
            (3) Statement I is correct but Statement II is incorrect
```



```
json_data_CHEMISTRY = [

        {
            "id": 1,
            "type": "CHEMISTRY",
            "question": """
            Given below are two statements: one is labelled as Assertion (A) and the other is labelled as Reason (R).
            Assertion (A): ICI is more reactive than I2.
            Reason (R): I-CI bond is weaker than I-I bond.
            In the light of the above statements, choose the most appropriate answer from the options given below:
            (1) Both (A) and (R) are correct and (R) is the correct explanation of (A).
            (2) Both (A) and (R) are correct but (R) is not the correct explanation of (A).
            (3) (A) is correct but (R) is not correct
            (4) (A) is not correct but (R) is correct
            """,
            "generated_answer": None,
            "correct_answer": 1
        },
        {
            "id": 2,
            "type": "CHEMISTRY",
            "question": """
            The IUPAC name of an element with atomic number 119 is
            (1) ununennium (2) unnilennium (3) unununnium (4) ununoctium
            """,
            "generated_answer": None,
            "correct_answer": 1
        },
        {
            "id": 3,
            "type": "CHEMISTRY",
            "question": """
            Match List-I with List-II.
            List-I (Drug class)
            (a) Antacids (b) Antihistamines (c) Analgesics (d) Antimicrobials
            List-II
            (Drug molecule)
            (i) Salvarsan (ii) Morphine (iii) Cimetidine (iv) Seldane
            Choose the correct answer from the options given below :
            (1) (a) - (iii), (b) - (ii), (c) - (iv), (d) - (i)          (2) (a) - (iii), (b) - (iv), (c) - (ii), (d) - (i)
            (3) (a) - (i), (b) - (iv), (c) - (ii), (d) - (iii)          (4) (a) - (iv), (b) - (iii), (c) - (i), (d) - (ii)
            """,
            "generated_answer": None
```



```
json_data_PHYSICS = [

    {
        "id": 1,
        "type": "PHYSICS",
        "question": """
        Match List-I with List-II
        List-I (Electromagnetic waves) : (a) AM radio waves (b) Microwaves  (c) Infrared radiations (d) X-rays
        List-II (Wavelength) (i) 10^-10 m (ii) 10^2 m (iii) 10^-2 m (iv) 10^-4 m
        Choose the correct answer from the options given below
        (1) (a) - (iv), (b) - (iii), (c) - (ii), (d) - (i)
        (2) (a) - (iii), (b) - (ii), (c) - (i), (d) - (iv)
        (3) (a) - (iii), (b) - (iv), (c) - (ii), (d) - (i)
        (4) (a) - (ii), (b) - (iii), (c) - (iv), (d) - (i)
      """,
        "generated_answer": None,
        "correct_answer": 4
    },
    {
        "id": 2,
        "type": "PHYSICS",
        "question": """
        The angular speed of a fly wheel moving with uniform angular acceleration changes from 1200 rpm to 3120
        rpm in 16 seconds. The angular acceleration in rad/s2 is
        (1) 2π
        (2) 4π
        (3) 12π
        (4) 104π
        """,
        "generated_answer": None,
        "correct_answer": 2
    },
    {
        "id": 3,
        "type": "PHYSICS",
        "question": """
        As the temperature increases, the electrical resistance
        (1) Increases for both conductors and semiconductors
        (2) Decreases for both conductors and semiconductors
        (3) Increases for conductors but decreases for semiconductors
        (4) Decreases for conductors but increases for semiconductors
      """,
```

1. Question Input:

   The answer_neet variable stores the result of calling the answer function with a specific NEET MCQ question related to half-wave rectification. The question asks about the output frequency in half-wave rectification when the input frequency is 60 Hz.

2. Function Execution:

   a. The answer function processes the input question and retrieves relevant documents from the physics domain using the PHY_retriever.

   b. It extracts the context from the relevant documents to provide background information for generating an answer.

3. LLM Chain Processing:
    a. The function uses the qa_chain_gpt LLM chain (GPT-3.5-Turbo) to generate an answer based on the extracted context and the given question.
    b. If the LLM chain is unable to provide a satisfactory answer ('SORRY' response), the function switches to the rag_free_chain to ensure a meaningful response.
4. Output: The result of the answer function, stored in answer_neet, is printed to display the generated answer to the NEET MCQ question about half-wave rectification's output frequency.

Now we’ll be creating and updating a JSON dataset containing physics-related questions with generated answers using the answer function previously defined. Here's a detailed explanation of the process:

1. JSON Data Initialization:
   The json_data_PHYSICS variable stores a list of dictionaries, each representing a physics question along with its ID, type, question content, generated answer (initially set to None), and the correct answer.
2. Update Function Initialization: The update_json_with_answers function takes two arguments: json_data (the JSON dataset) and answer_function (the function used to generate answers for questions).
3. Iterating Through Questions:
    a. The code iterates through each element (question dictionary) in the json_data_PHYSICS list.
    b. It extracts the question content from each element.
4. Generating Answers:
    a. The answer_function is called with each question to generate a response based on the question's content and context.
    b. The generated_answer variable stores the result returned by the answer_function.
5. Updating JSON Data:

   a. The generated_answer is then assigned to the "generated_answer" key in the corresponding question element within the JSON dataset.
   b. This update ensures that each question in the JSON dataset now includes a generated answer.
6. Writing Updated Data to JSON File:
   a. The updated_data variable stores the JSON dataset with generated answers.
   b. The updated JSON data is then written to a file named physics_json.json using the json.dump() function, with proper formatting (indent=4) for readability.
7. Final Output:
   After execution, physics_json.json contains the original physics questions along with their correct answers and newly generated answers, making it a comprehensive dataset for evaluation and analysis.

This code segment automates the process of generating and updating answers for physics questions in a JSON format, enabling efficient handling and management of question-answer pairs for further analysis or application in educational or testing scenarios.

After executing the code to generate answers for physics questions and updating the JSON dataset with these answers, we obtain a JSON file that serves as a comprehensive repository of questions, their IDs, subject types, generated answers (including the selected option and the reasoning/explanation), and the correct answers. This process lays the foundation for a detailed manual verification of each generated answer against the corresponding correct answer within the JSON file.
Manual verification involves carefully examining each question-answer pair to ensure the accuracy and correctness of the generated answers. This meticulous process is crucial for evaluating the performance and reliability of the LLM model in accurately responding to the given questions.

The first step in manual verification is to open the generated JSON file containing the updated data. Each question, along with its generated answer and correct answer, is reviewed systematically. For each question:

1. Comparison of Generated and Correct Answers:
    a. The generated answer, comprising the selected option and the accompanying explanation or reasoning, is compared with the correct answer provided in the JSON file.
    b. Attention is paid to details such as the correctness of the selected option, the accuracy of the explanation or reasoning provided, and the alignment with the expected correct answer.
2. Evaluation of Explanation/Reasoning:
    a. The explanation or reasoning provided alongside the generated answer is scrutinized to ensure its relevance, clarity, and completeness in justifying the selected option.
    b. Factors such as logical reasoning, factual accuracy, and coherence with the question context are considered during this evaluation.
3. Recording Discrepancies or Inconsistencies:
    a. Any discrepancies or inconsistencies between the generated answer and the correct answer are noted down for further analysis.
    b. This includes cases where the generated answer is incorrect, lacks sufficient explanation, or deviates significantly from the expected correct answer.
4. Documentation and Analysis:
    a. Detailed documentation of the verification process is maintained, including observations, findings, and any patterns or trends noticed during the comparison.
    b. Analysis of the verification results helps in identifying areas of improvement, evaluating the model's performance, and refining the question-answer generation process.

By meticulously checking each generated answer against the correct answer manually, we ensure the quality and reliability of the LLM model's responses, contributing to the overall accuracy and effectiveness of the question-answering system.

Implementation for the other two subjects i.e. PHYSICS and BIOLOGY:

For the NEET question-answering system, extending beyond physics to include subjects like chemistry and biology involves a systematic approach similar to what we implemented for physics. The process begins by creating separate directories for each subject to organize the respective documents and data. In this case, we create directories named 'CHEM' for chemistry-related documents and 'BIO' for biology-related documents within the Colab session storage.

The next step is to upload the relevant documents for chemistry and biology into their respective directories. This involves sourcing authentic and comprehensive study materials specific to chemistry and biology subjects, ensuring a diverse range of topics and question types representative of the NEET exam syllabus. The documents can include textbooks, study guides, practice papers, and other educational resources relevant to chemistry and biology.

Once the documents are uploaded, we proceed with the document ingestion process, where we load and chunk the documents to prepare them for analysis and retrieval. This step involves utilizing tools and techniques to break down large documents into smaller, manageable chunks, facilitating efficient processing and retrieval during the question-answering process. The loaded and chunked documents are then stored in the respective directories ('CHEM' and 'BIO') for further processing.

Following document ingestion, we create subject-specific vector databases for chemistry and biology using the loaded and chunked documents. This involves encoding the document chunks into vector representations using embedding models like GoogleGenerativeAIEmbeddings for chemistry and biology subjects. The vector databases are structured to enable efficient retrieval and processing of information during question answering, enhancing the system's accuracy and performance in generating well-informed responses.

The next crucial step is initializing the LLM chains and prompt templates specific to chemistry and biology subjects. Similar to what we did for physics, we create LLM chains for chemistry ('qa_chain_chem') and biology ('qa_chain_bio'), each tailored to handle questions and generate answers relevant to their respective subjects. Additionally, we define prompt templates that guide the LLMs in understanding and processing user queries related to chemistry and biology topics.

We then execute the answer function using the LLM chains created for chemistry and biology, retrieving and generating answers for NEET multiple-choice questions in chemistry and biology subjects. The answers are compared against the correct answers stored in the JSON dataset, ensuring the accuracy and reliability of the generated responses.

Finally, we update the JSON dataset with the generated answers for chemistry and biology questions, following the same process of manual verification to check each generated answer against the correct answer. This meticulous verification process ensures the quality and consistency of the question-answering system across different subjects, providing accurate and informative responses to NEET exam-related queries in chemistry, biology, and physics.

6.2 Results

In the context of our research study, the results were calculated by checking our predicted answer against the ground truth available in the NEET answer booklet. For each multiple choice question with 4 options, a single correct choice was available. To evaluate the results, we have taken inspiration from the standard ML metric - accuracy and tweaked it from the conventional machine learning formula typically used in evaluating machine learning models, because we have a deterministic (i.e. single-class) prediction. While the standard accuracy formula for binary class prediction incorporates true positives, true negatives, false positives, and false negatives, our approach focuses on a simpler metric: the ratio of true positives i.e. total correct answers to total questions, expressed as a percentage, as we do not possess the false negative values. As NEET penalizes incorrect answers, we have included precision, where incorrect answers are considered as false positives.

Our equations for accuracy and precision are as follows:

$$Accuracy = \frac{True\ Positives}{Total\ questions}$$

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

By defining accuracy as the ratio of correct answers to total questions attempted, we aimed to provide a straightforward measure of the model's performance relative to the questions presented. This metric allows for a direct assessment of the model's effectiveness in answering NEET exam-style questions without the need for complex evaluation formulas that may not align with the nature of LLM-based systems.

Moreover, it's important to note that this research marks the first attempt at evaluating the performance of LLM+RAG models in the context of NEET question answering systems, and as such, there is no official comparison available for reference. However, insights from previous experiments based solely on LLMs, such as those reported by Analytics India Magazine[8], indicate that models like ChatGPT achieved a performance level of approximately 50% in similar assessments, with poor scores in botany because of the unavailability of precise knowledge.

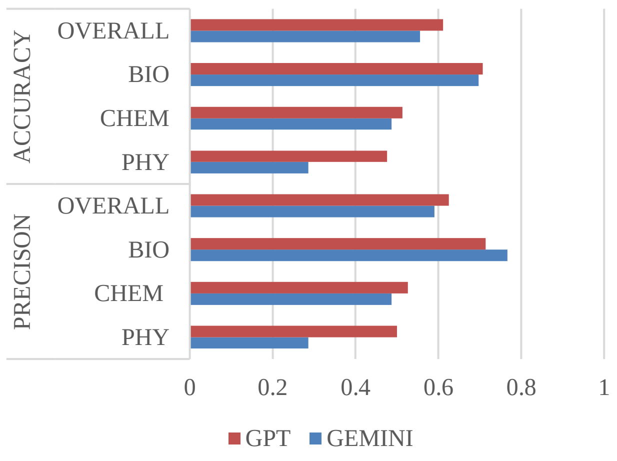
Table displays the outcomes of the evaluation of our answer generation phase. The results from both models show a close comparison, with GPT slightly outperforming. Notably, the GPT-based QA system managed to achieve precision rates of 50% and above for all three sections and a 62.5% overall precision. Particularly in Physics, GPT not only provided superior answers but also abstained from hunching to prevent incorrect responses. Remarkably, both models excelled in Biology questions, achieving an accuracy and precision rate of nearly 70%, a significant improvement compared to GPT's previous performance in this subject [8].On the other hand, the overall performance QA system, especially for Gemini was below average in physics, which had higher number of numerical and application based questions.

The LLM models operate majorly on probabilities, generating responses based on their pre-trained knowledge. Evaluating LLM performance using traditional accuracy metrics may not accurately reflect its capabilities or limitations. Hence, with our results, we do not assess the LLMs but the question-answering system developed over the LLMs. LLMs, particularly when augmented with RAG, exhibit nuanced behavior influenced by factors such as the quality and relevance of external knowledge sources, the complexity of the queries, and the model's inherent biases

Table I: Results for NEET 2022 Q1 Paper

		Correct	Wrong	Unanswered
GEMINI	PHY	12	30	0
	CHEM	19	20	0
	BIO	69	21	9
	OVERALL	100	71	9
GPT	PHY	20	20	2
	CHEM	20	18	1
	BIO	70	28	1
	OVERALL	110	66	4

and limitations. In this particular use case, LLM’s decisive power can be tested by understanding the patterns of confidence level of predicted answer for a particular question.



Chapter 7

Conclusion and Future work

7.1 Conclusion

Our research delves into the intricacies of developing a question-answering system leveraging Large Language Models (LLMs) integrated with Retrieval-Augmented Generation (RAG) techniques, particularly within the specialized domain of NEET (National Eligibility-cum-Entrance Test). By harnessing external knowledge repositories such as NCERT textbooks and employing innovative methodologies like context-aware chunking and similarity-based retrieval, our proposed RAG framework exhibits promising performance metrics—albeit not reaching human-like levels—in terms of accuracy and relevance when compared to traditional LLM approaches.

Through rigorous experimentation, our system achieved an impressive accuracy and precision rate of approximately 60%, showcasing notable proficiency, particularly in the challenging domain of Biology, where previous LLM-based approaches struggled to provide satisfactory responses. This underscores the significance of augmenting LLMs with external knowledge sources to surmount inherent limitations such as outdated information and lack of domain expertise.

However, our study also uncovered several challenges inherent in RAG systems, including the need for pre-processing techniques like query simplification, advanced retrieval methods, and post-retrieval processing. These challenges, while posing hurdles to the model's performance, serve as crucial focal points for future research endeavors and provide valuable insights for designing more effective RAG-based chatbots.

In conclusion, our study makes a meaningful contribution to the evolving landscape of research on integrating external knowledge sources with LLMs, offering valuable insights into the potential applications and advantages of RAG techniques in specialized domain contexts. Through our experimentation and analysis, we pave the way for further advancements in question-answering systems, with the ultimate goal of enhancing user experiences and knowledge dissemination in critical domains like medical education.

7.2 Future Scope

1. In specialized exams like NEET, questions are often designed to test not only factual knowledge but also the ability to interpret complex scenarios and apply critical thinking skills. As a result, many questions may contain layers of meaning or require careful analysis to uncover the underlying intent. Semantic analysis plays a crucial role in deciphering such questions. By breaking down the query appropriately, we can extract the core elements of the question and gain a deeper understanding of its nuances. This involves parsing the text to identify key words and phrases, examining the structure of the sentence, and considering contextual clues that may provide additional insight. For example, consider a question that asks about the physiological effects of a certain medication on the human body. At first glance, the question may appear straightforward, but upon closer inspection, it becomes clear that multiple layers of analysis are required. Semantic analysis helps us dissect the question, identify the specific medication and its physiological effects, and discern any underlying assumptions or contextual information that may impact the answer. In the context of our system, pre-processing queries involves implementing sophisticated algorithms and techniques to handle such complex questions effectively. This may include natural language processing (NLP) tools for semantic analysis, syntactic parsing to identify sentence structure, and machine learning models trained on specialized domain data to improve accuracy and relevance in question interpretation. By investing in robust pre-processing capabilities, we ensure that our system can accurately capture the essence of each question, leading to more precise and reliable responses.
2. Optimizing retrievers: In the discussion section, we highlighted a limitation related to the effectiveness of similarity search in retrieving relevant

information in domains like physics. To address this limitation, alternative search methods, such as keyword search, can be explored. Unlike similarity search, which relies on content similarity, keyword search can effectively retrieve relevant formulas and concepts based on specific keywords or terms present in the query while considering academic question-answering systems. Moreover, a combination of different searches in retrieval methods can assist in providing accurate information to the LLM chain.

3. Post-processing relevant documents: Summarization or contextual refinement, can be applied to the retrieved chunks to distill the information and extract the required data from most relevant retrieved content. This post-processing step can help streamline the information retrieved from the corpus, making it more concise and aligned with the user query. By integrating these alternative search methods and post-processing techniques, the system can enhance its ability to retrieve and present relevant information, particularly in domains characterized by complex, indirect, and redundant context.
4. Integration of Vision Language Models (VLMs): Incorporating VLMs into the RAG framework will enable the system to handle questions containing images and equations. By leveraging VLMs, the model can interpret visual information, such as diagrams, charts, and graphs, commonly found in educational materials like NCERT textbooks. This enhancement will broaden the scope of questions that the system can effectively answer, providing more comprehensive assistance to users preparing for exams like NEET.
5. Enhanced Multimodal Capabilities: Beyond images and equations, future iterations of RAG can explore incorporating various methods to extract tables, complex equations, diagrams, and other visual representations, along with methods to map them into vector databases. By seamlessly integrating these diverse forms of information, the system can offer a more immersive and interactive educational environment, catering to diverse learning preferences and enhancing comprehension.
6. Domain Expansion and Specialization: Expanding the application of RAG by using vivid study resources for NEET and combining multiple chunks for cross topic knowledge bank can assist in answering questions having multiple layers of information.

References

[1] H. Jung *et al.*, “Enhancing Clinical Efficiency through LLM: Discharge Note Generation for Cardiac Patients,” pp. 1–10, 2024, [Online]. Available: http://arxiv.org/abs/2404.05144

[2] H. Zhao *et al.*, “Revolutionizing Finance with LLMs: An Overview of Applications and Insights,” pp. 1–37, 2024, [Online]. Available: http://arxiv.org/abs/2401.11641

[3] E. Haaralahti, “Utilization of local large language models for business applications,” 2024.

[4] S. M. Kelly, “ChatGPT passes exams from law and business schools,” *CNN Business*, 2023. [Online]. Available: https://edition.cnn.com/2023/01/26/tech/chatgpt-passes-exams/index.html

[5] J. Lubell, “ChatGPT passed the USMLE. What does it mean for med ed?,” *American Medical Association(AMA)*, 2023. [Online]. Available: https://www.ama-assn.org/practice-management/digital/chatgpt-passed-usmle-what-does-it-mean-med-ed

[6] L. Varanasi, “ChatGPT is on its way to becoming a virtual doctor, lawyer, and business analyst. Here’s a list of advanced exams the AI bot has passed so far.,” Business Insider. [Online]. Available: https://www.businessinsider.in/tech/news/chatgpt-is-on-its-way-to-becoming-a-virtual-doctor-lawyer-and-business-analyst-hereaposs-a-list-of-advanced-exams-the-ai-bot-has-passed-so-far-/slidelist/97388435.cms

[7] D. Sharma, “ChatGPT passed Wharton’s MBA but failed UPSC prelims: here is what you need you know,” *India Today*, New Delhi, Mar. 06, 2023. [Online]. Available: https://www.indiatoday.in/technology/news/story/chatgpt-passed-whartons-mba-but-failed-upsc-prelims-here-is-what-you-need-you-know-2343171-2023-03-06

[8] S. Saha, “ChatGPT Takes NEET; Will it Pass with Flying Colors or Flunk?,” AI Trends & Future. [Online]. Available: https://analyticsindiamag.com/chatgpt-takes-neet-will-it-pass-with-flying-colors-or-flunk/

[9] Dr. Sajjad Mahmood, “Understanding the Limitations of Language Models.” [Online]. Available: https://www.linkedin.com/pulse/understanding-limitations-language-models-dr-sajjad-mahmood-zslsf/

[10] E. Kasneci *et al.*, “ChatGPT for good? On opportunities and challenges of large language models for education,” *Learn. Individ. Differ.*, vol. 103, p. 102274, Apr. 2023, doi: 10.1016/J.LINDIF.2023.102274

[11] Y. Gao *et al.*, “Retrieval-Augmented Generation for Large Language Models: A Survey,” Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.10997

[12] X. Du and H. Ji, “Retrieval-Augmented Generative Question Answering for Event Argument Extraction,” *Proc. 2022 Conf. Empir. Methods Nat. Lang. Process. EMNLP 2022*, no. 1, pp. 4649–4666, 2022, doi: 10.18653/v1/2022.emnlp-main.307

[13] P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” *Adv. Neural Inf. Process. Syst.*, vol. 2020-Decem, no. NeurIPS, 2020.

[14] Z. Levonian *et al.*, “Retrieval-augmented Generation to Improve Math Question-Answering: Trade-offs Between Groundedness and Human Preference,” Oct. 2023, [Online]. Available: http://arxiv.org/abs/2310.03184

[15] C. Wu, W. Lin, X. Zhang, Y. Zhang, Y. Wang, and W. Xie, “PMC-LLaMA: Towards Building Open-source Language Models for Medicine,” 2023, [Online]. Available: http://arxiv.org/abs/2304.14454

[16] M. Kulkarni, P. Tangarajan, K. Kim, and A. Trivedi, “Reinforcement Learning for Optimizing RAG for Domain Chatbots,” Jan. 2024, [Online]. Available: http://arxiv.org/abs/2401.06800

[17] Z. Xu *et al.*, “ActiveRAG: Revealing the Treasures of Knowledge via Active Learning,” 2024, [Online]. Available: http://arxiv.org/abs/2402.13547

[18] G. Xiong, Q. Jin, Z. Lu, and A. Zhang, “Benchmarking Retrieval-Augmented Generation for Medicine,” 2024, [Online]. Available: http://arxiv.org/abs/2402.13178

[19] J. Lála, O. O’Donoghue, A. Shtedritski, S. Cox, S. G. Rodriques, and A. D. White, “PaperQA: Retrieval-Augmented Generative Agent for Scientific Research,” Dec. 2023, [Online]. Available: http://arxiv.org/abs/2312.07559

[20] H. Liu *et al.*, “Retrieval augmented scientific claim verification,” *JAMIA Open*, vol. 7, no. 1, 2024, doi: 10.1093/jamiaopen/ooae021

[21] *Textbooks PDF*. [Online]. Available: https://ncert.nic.in/textbook.php

[22] *NEET (UG) Notes Biology*. Oswal Publications. [Online]. Available: https://oswaalbooks.com/pages/neet-ug-notes-biology

[23] H.C Verma, *Concepts of Physics*. [Online]. Available: https://hcverma.in/books