Enhanced RAG (Retrieval-Augmented Generation) Based Chatbot For Educational Institutions

Dissertation-II

Submitted in partial fulfillment of the requirements for the degree of

Master of Computer Applications

in

Department of Computer Applications

by

Harshita Sharma

23MCA0069

Under the guidance of

Dr. Karthikeyan J

School of Computer Science Engineering and Information Systems

VIT, Vellore

April, 2025

Enhanced RAG (Retrieval-Augmented Generation) Based Chatbot For Educational Institutions

Dissertation-II

Submitted in partial fulfillment of the requirements for the degree of

Master of Computer Applications

in

Department of Computer Applications

by

Harshita Sharma

23MCA0069

Under the guidance of

Dr. Karthikeyan J

School of Computer Science Engineering and Information Systems

VIT, Vellore

April, 2025

DECLARATION

I hereby declare that the **PMCA699J - Dissertation-II** entitled "Enhanced RAG (Retrieval-Augmented Generation) Based Chatbot For Educational Institutions" submitted by me, for the award of the degree of *Master of Computer Applications in Department of Computer Applications, School of Computer Science Engineering and Information Systems* to VIT is a record of bonafide work carried out by me under the supervision of Dr. Karthikeyan J, Professor Grade 1, SCORE, VIT, Vellore.

I further declare that the work reported in this dissertation has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date:

Signature of the Candidate

CERTIFICATE

This is to certify that the **PMCA699J - Dissertation-II** entitled "Enhanced RAG (Retrieval-Augmented Generation) Based Chatbot For Educational Institutions" submitted by **Harshita Sharma & 23MCA0069**, SCORE, VIT, for the award of the degree of *Master of Computer Applications in Department of Computer Applications*, is a record of bonafide work carried out by him under my supervision during the period, 13. 12. 2024 to 17.04.2025, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The dissertation fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place: Vellore
Date:

Signature of the VIT-SCORE - Guide

Internal Examiner **External Examiner**

Head of the Department
Department of Computer Application

ACKNOWLEDGEMENTS

It is my pleasure to express with a deep sense of gratitude to my **Dissertation-II** guide Dr. **Karthikeyan J, Professor Grade 1** School of Computer Science Engineering and Information Systems, Vellore Institute of Technology, Vellore for his constant guidance, continual encouragement, in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part to work with an intellectual and an expert in the field of Natural Language Processing.

"I would like to express my heartfelt gratitude to Honorable Chancellor **Dr. G Viswanathan**; respected Vice Presidents **Mr. Sankar Viswanathan**, **Dr. Sekar Viswanathan**, Vice Chancellor **Dr. V. S. Kanchana Bhaaskaran**; Pro-Vice Chancellor **Dr. Partha Sharathi Mallick**; and Registrar **Dr. Jayabarathi T**.

My whole-hearted thanks to Dean **Dr. Daphne Lopez**, School of Computer Science Engineering and Information Systems, Head, Department of Computer Applications **Dr. E Vijayan**, MCA Project Coordinator **Dr. Karthikeyan P**, SCORE School Project Coordinator **Dr. Thandeeswaran R**, all faculty, staff and members working as limbs of our university for their continuous guidance throughout my course of study in unlimited ways.

It is indeed a pleasure to thank my parents and friends who persuaded and encouraged me to take up and complete my dissertation successfully. Last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly towards the successful completion of the dissertation.

Place: Vellore

Date: **Harshita Sharma**

Executive Summary

This paper presents the development of an Enhanced RAG-powered chatbot designed specifically for educational institutions. Building upon the initial naive implementation, this version incorporates advanced yet simple-to-implement techniques to improve accuracy, efficiency, and response quality. The development process followed an iterative approach, where each advanced technique was first studied in isolation and then combined strategically to determine the most effective configuration. Motivated by feedback from the previous version, the primary focus was on enhancing response accuracy and contextual framing. Key improvements include Query Decomposition to refine user intent understanding, Reciprocal Rank Fusion (RRF) re-ranking to retrieve only the most relevant context, and Prompt Engineering to generate clearer, more structured responses. Additionally, expanded domain-specific data was incorporated, covering university courses, hostels, food points, and other student-relevant information. By prioritizing practical and efficient enhancements over unnecessary complexity, the chatbot now delivers more precise and context-aware answers, making it a valuable tool for educational institutions. Future work will explore multi-turn conversations, real-time document updates, advanced indexing techniques, and AI agent integration to further optimize its capabilities.

CONTENTS

List of Figures

List of Tables

List of Abbreviations

RAG	Retrieval Augmented Generation
LLM	Large Language Models
RRF	Reciprocal Rank Fusion
ICL	In-Context Learning
CoT	Chain-of-Thought
SSR	Step-By-Step Reasoning
ToT	Tree-of-Thought

CHAPTER 1

INTRODUCTION

The rapid evolution of Large Language Models (LLMs) has significantly transformed the field of Natural Language Processing (NLP), enabling machines to interpret and generate human-like text with remarkable precision. Trained on vast and diverse datasets, these models can efficiently perform a variety of tasks, including question answering, summarization, and content creation. However, despite their impressive capabilities, LLMs have inherent limitations. They rely heavily on static training data, which can quickly become outdated or insufficient for domain-specific or real-time queries. Additionally, LLMs are prone to hallucination, often generating inaccurate or fabricated information, making them unreliable for fact-based applications. The introduction of ChatGPT in 2022 marked a turning point in the technology landscape, showcasing the power of LLMs on a massive scale. Its remarkable ability to engage in natural conversations, generate creative content, and assist with complex queries fueled a widespread surge of interest in Artificial Intelligence (AI). Although chatbots had existed for years, ChatGPT's unprecedented success reshaped the industry, inspiring developers and organizations to explore AI-powered solutions for various applications. By 2024, nearly every major tech company had either adopted AI to automate routine tasks or was actively developing AI-driven solutions to enhance operational efficiency.

As organizations increasingly embraced AI, the need for custom, domain-specific chatbots became evident. While generalized LLMs like ChatGPT are powerful, they often struggle with delivering accurate, context-specific responses due to their reliance on static, broad-spectrum training data. This highlighted the necessity of developing tailored AI models capable of accessing real-time, domain-specific knowledge.

To address these challenges, Retrieval-Augmented Generation (RAG) emerged as a powerful technique. RAG enhances LLMs by integrating dynamic information retrieval from external knowledge sources, such as vector databases or document repositories, significantly improving the accuracy and relevance of responses. This makes RAG-based systems particularly effective for applications like chatbots, where delivering precise, up-to-date information is essential. Many organizations have already started leveraging RAG-powered chatbots to offer more accurate and context-aware answers, making them highly valuable for industries such as education, healthcare, and customer service. With the growing demand for reliable, domain-specific AI solutions, RAG represents a significant step forward in overcoming the limitations of traditional LLMs and enhancing the practical utility of AI-powered systems.

1.1 Objective

This project focuses on developing a Enhanced RAG-powered chatbot tailored for the education domain, designed to answer student queries in real time with improved accuracy and contextual relevance. The first prototype of this chatbot was a naive implementation of RAG, deployed to gather feedback on its performance with actual student queries. By analyzing the collected responses, valuable insights were obtained regarding the chatbot's accuracy, efficiency, and the types of questions students frequently ask. This analysis highlighted the limitations of the naive RAG implementation, paving the way for further enhancements to improve the chatbot's reliability and overall performance.

1.2 Motivation

The integration of technology across various domains has significantly enhanced their growth and efficiency, including the education sector. It has been observed that industries and sectors that evolve alongside technological advancements consistently achieve exceptional results by maximizing the benefits of innovation. The same holds true for education. The use of chatbots initially gained traction when large companies started deploying them to handle customer queries. The notable improvement in customer satisfaction delivered by these AI-powered bots highlighted their potential, leading to their adoption across multiple domains, including education.

When applying for admission to a new university or school, students and parents often have numerous questions and doubts that typically require assistance from human faculty or staff. Although multiple online platforms now provide information about universities, and students frequently share their experiences on social media, the demand for dedicated virtual assistants remains evident. With thousands of students seeking admission each year, the volume of queries increases significantly. Rather than burdening human faculty with addressing repetitive or straightforward inquiries, AI-powered chatbots can efficiently handle such tasks, allowing educators to focus on more critical and complex matters.

Motivated by this need, I decided to develop a RAG-based chatbot specifically for educational institutions. While most universities already have official websites and platforms containing all the necessary information, students often find these websites difficult to navigate. This chatbot aims to bridge that gap by offering a user-friendly, real-time solution to answer student queries accurately and efficiently.

1.3 Background

The rapid advancement of Generative AI (GenAI) has significantly transformed the field of artificial intelligence, enabling machines to autonomously generate content such as text, images, and even code. GenAI models, including large language models (LLMs) like GPT, Gemini, and Claude, are trained on vast datasets encompassing diverse domains, allowing them to perform tasks such as content generation, summarization, and question answering with remarkable accuracy. In recent years, GenAI-powered applications have become increasingly prevalent across various industries, automating repetitive tasks and enhancing productivity. In customer service, AI-powered chatbots handle frequently asked questions (FAQs), reducing the workload on human agents. Similarly, in education, GenAI is being leveraged to assist students with administrative queries, provide real-time information, and offer personalized learning experiences. The RAG chatbot developed in this project is a GenAI application specifically designed for the educational domain. By combining LLM-based content generation with external context retrieval from a vector database, the chatbot provides accurate and contextually relevant answers to student queries. This integration enables it to overcome the limitations of static training data, ensuring that the responses remain up-to-date and domain-specific.

CHAPTER 2

DISSERTATION DESCRIPTION AND GOALS

2.1 Project Description

This project focuses on the development of an Enhanced Retrieval-Augmented Generation (RAG) chatbot designed specifically for the education domain, aiming to answer student queries in real time with improved accuracy and contextual relevance. The chatbot is intended to assist students by providing instant and reliable information related to admissions, courses, hostel fees, faculty details, and other administrative queries. The development process began with the deployment of a first prototype featuring a naive RAG implementation. This initial version was released to gather real-world feedback by exposing the chatbot to actual student queries. The feedback analysis provided valuable insights into the chatbot's performance, highlighting accuracy gaps, inefficiencies, and limitations in handling specific types of questions. It also revealed which stages in the RAG pipeline required refinement to enhance the model's overall effectiveness.

Based on the feedback, the project entered the enhancement phase, incorporating advanced RAG techniques to improve the retrieval accuracy, ranking effectiveness, and contextual generation of the chatbot. The prototype's limitations guided the targeted application of these techniques to specific stages in the pipeline, addressing the issues identified during the evaluation phase. To enhance the chatbot's knowledge coverage, the dataset was expanded to include detailed information on courses, departmental details, hostel fees, common student queries, and block-related data. This comprehensive dataset ensures that the chatbot can provide precise and relevant answers to a wider range of student inquiries. The resulting enhanced RAG chatbot offers greater reliability, efficiency, and contextual accuracy, making it a valuable tool for educational institutions. Even though the dataset or the documents provided to the chatbot belong to VIT, but the chatbot works the same for any other educational institution. It streamlines repetitive tasks, improves information accessibility, and enhances the overall student experience by providing instant and consistent assistance.

2.2 Goals

The project aims to develop an enhanced version of the previously implemented naive RAG chatbot by incorporating various advanced techniques into the relevant phases of the RAG pipeline. The current version of the chatbot employs Query Decomposition to effectively handle complex user queries, enabling the model to better interpret and understand multi-faceted questions. Additionally, re-ranking techniques have been applied, specifically using the widely recognized Reciprocal Rank Fusion (RRF) method. This technique re-ranks the retrieved documents, ensuring that only the most relevant and highest-ranked context is passed to the LLM for generating responses. To further enhance the chatbot's output quality, prompt engineering has been implemented, refining how the LLM formulates and generates answers. These enhancements have significantly improved the accuracy, coherence, and contextual relevance of the chatbot's responses. To demonstrate the effectiveness of these advanced modifications, a comparative table has been included, showcasing the differences in the chatbot's performance before and after the enhancements. The table highlights improvements in query interpretation and answer generation. Beyond these technical upgrades, the chatbot's knowledge base has been expanded by adding additional documents. This includes comprehensive syllabus of all courses under the SCORE department at VIT, hostel fee structures for both men and women hostels, block information, as well as details about shops and food points on campus. This enriched dataset ensures the chatbot can handle a broader range of student queries with greater accuracy and detail.

CHAPTER 3

TECHNICAL SPECIFICATIONS

- Programming Language
    - Python 3.12 for chatbot development
- Libraries and Frameworks
    - Natural Language Processing: Langchain for text processing and language understanding
    - Embedding Model: Google Generative AI Embedding for document

vectorization.

- Data Storage
  - ChromaDB, a vector store for storing vector embeddings.
- Environment
  - Code editor and Development Environment: Visual Studio Code as code editor and conda to manage python dependencies.

CHAPTER 4
DESIGN APPROACH AND DETAILS

4.1 Design Approach

The initial design approach for the chatbot involved building an RAG-powered system using LangChain, a robust framework for developing LLM-based applications. Since data quality and coverage play a crucial role in ensuring the accuracy and reliability of any AI application, I utilized university documents in PDF format as the primary data source. To process these PDFs, I used PyPDF Reader, a powerful library for handling PDF files, to extract the content and convert it into plain .txt files. For document embedding, I opted for Google Gemini Embedding, as the task required natural language understanding and context-aware text generation. Unlike traditional NLP preprocessing techniques (e.g., tokenization, stemming, or stop-word removal), I relied on the embedding model's capabilities to transform the textual content into meaningful vector representations. During this process, the embedding model itself handles text formatting and cleaning, ensuring consistent and high-quality vector conversions. To store the vector embeddings, ChromaDB, an efficient vector database is used, enabling fast and reliable retrieval of relevant documents. During the query phase, the user’s input is converted into vector format, and the most relevant documents are retrieved by identifying the closest matching vectors. These retrieved documents and user’s input is given to the LLM, which in turn generates the appropriate response.

The fundamental working of the enhanced chatbot remains the same as the previous version; however, the difference lies in how each stage of the RAG pipeline has been refined with advanced techniques where necessary. A RAG pipeline consists of multiple stages, with various techniques available for each stage. Since the chatbot is not a complex reasoning application but a context-based question-answering system, it simply needs to retrieve and provide accurate answers from the available documents. Therefore, implementing unnecessary advanced techniques would have resulted in an overly complex and bulkier application, making it harder to manage. To avoid this, I carefully analyzed the errors and performance bottlenecks in the initial prototype and identified the specific stages where improvements were required. Based on this analysis, I selected a combination of advanced techniques that best addressed the chatbot's limitations and effectively handled student queries. After practical experimentation, the most beneficial combination included:

- Query Decomposition – to better interpret and break down complex user queries.
- Re-ranking using Reciprocal Rank Fusion (RRF) – to retrieve and prioritize only the most relevant documents for the LLM.
- Prompt Engineering – to enhance the quality and clarity of the chatbot's responses by refining how the LLM generates answers.

4.2 Materials and Methods

4.2.1. Foundation

<u>Problem identification</u>

This enhanced version of the chatbot was developed by incorporating feedback from the previous version. During the analysis, I identified three major issues with the initial implementation:

1. Incomplete or poorly structured queries:
   When users entered incomplete questions or omitted certain keywords, the model failed to retrieve relevant documents or context. As a result, it either returned the message:

   - *"Answer not provided in the context"*

- or generated hallucinated responses.

2. Failure to retrieve complete document chunks:
   Since the documents were divided into smaller chunks, the model was unable to retrieve all the necessary chunks required to generate a comprehensive answer. This often led to incomplete or partially correct responses.

3. Inconsistent out-of-context responses:
   When asked questions outside the context, the chatbot provided random and inconsistent responses instead of a standardized, formatted message indicating that the query was out of scope.

<u>Enhancement in the current version</u>

To rectify these issues, various stages of the RAG pipeline were modified and improved by incorporating advanced techniques, as described earlier.

1. Query Rewriting with LLM:
   The model now begins by capturing the user's input. However, instead of directly embedding the query and retrieving documents, an LLM call is made to rewrite the user query into five different versions. This step ensures that the query is structured properly, making it easier for the model to understand and interpret the user's intent.

2. Multi-Query Retrieval and Aggregation:
   For each of the five reformulated queries, the most relevant documents are retrieved. These documents are then aggregated into a single collection, ensuring that no relevant information is missed.

3. Re-ranking for Higher Accuracy:
   To further refine the retrieval process, a unique re-ranking technique is applied. This technique calculates the rank of each document across all the reformulated queries and generates a list of the highest-ranked documents. This ensures that only the most

contextually relevant documents are used as input for the LLM.

4. Minimizing Irrelevant Context:
   By focusing only on the highly relevant documents, the model avoids providing excessive or redundant context, which could otherwise confuse the LLM and lead to inaccurate responses. This is similar to how, in machine learning, having too many irrelevant features can cause overfitting and reduce the model's performance.

5. Prompt Engineering for Accuracy:
   Finally, prompt engineering techniques are applied to enhance the LLM's response quality, ensuring that the answers are accurate, coherent, and properly formatted.

Below given are the system architectures of naive RAG and advanced RAG clearly showing the improvements at stages mentioned:

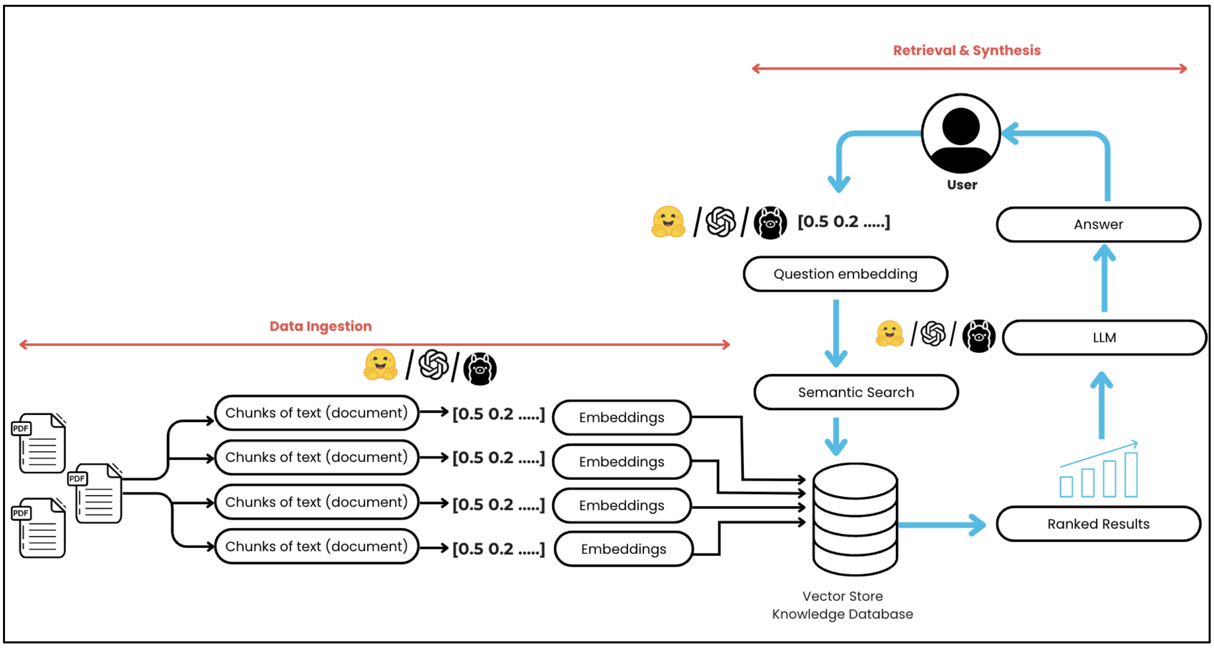

Fig. 4.1. Naive RAG Architecture

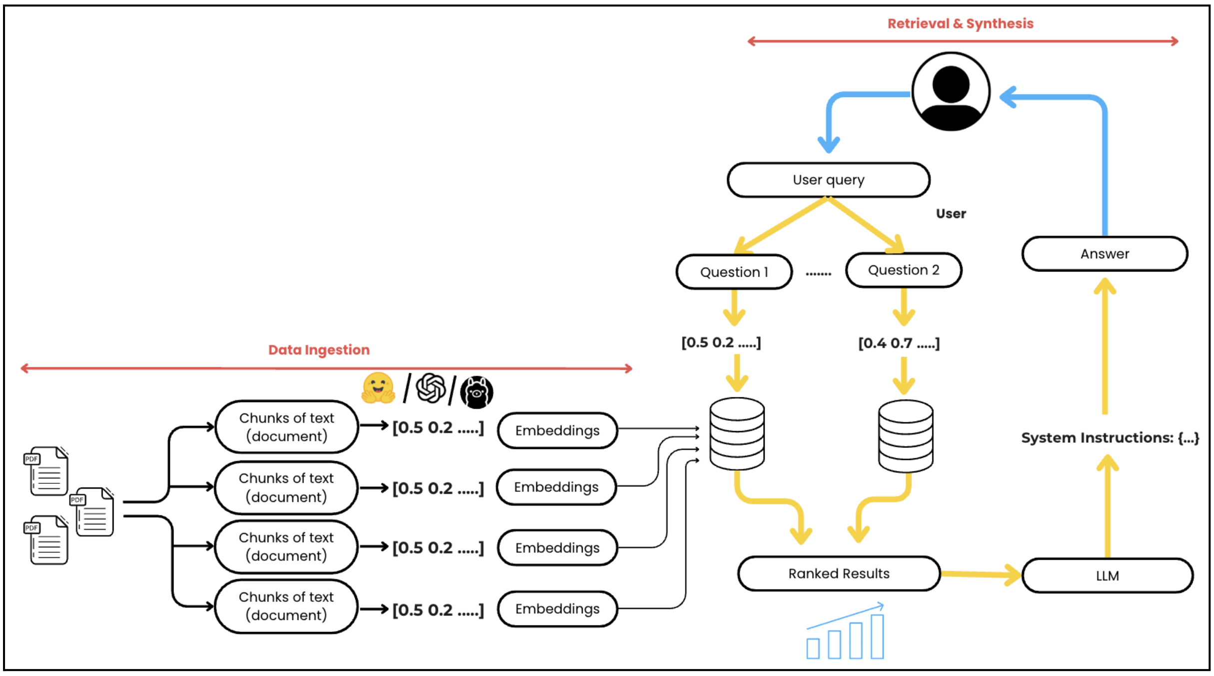

Fig. 4.2. Advanced RAG Architecture

In their paper, Siran Li et al. explore various key factors that influence the performance of Retrieval-Augmented Generation (RAG) systems. They provide a detailed analysis of how these factors impact the quality of generated responses and propose best practices for optimizing RAG models. Their study offers strategies to balance contextual richness with retrieval efficiency, ensuring more effective and accurate results. The authors formulate nine major research questions, beginning with whether the size of the Large Language Model (LLM) affects response quality and extending to inquiries such as whether focusing on a few retrieved sentences improves RAG's responses. Drawing insights from these questions, I examined several of them in the context of this chatbot project. In the subsequent sections of this paper, I discuss some of these factors in detail, including:

- Does prompting influence the alignment between retrieval and generation?
- How does retrieved chunk size impact response quality?

Additionally, I highlight the techniques implemented to enhance the chatbot's performance based on these findings.

4.2.2. Query Translation

Query translation is a technique that rewrites user queries into a more effective and optimized version to enhance search results. This process addresses common issues such as typos, ambiguous phrasing, synonym expansion, and restructuring to improve document retrieval accuracy. It is unrealistic to expect users to always input perfectly phrased queries. A major limitation identified in the naïve chatbot was its inability to retrieve relevant documents if the user's query was incomplete or incorrectly phrased. As a result, even when the required information existed in the vector store, the chatbot failed to generate accurate responses. To address this, various query translation techniques have been explored

Multi-Query Generation:

In this technique, an LLM generates multiple variations of the user's query, ensuring that all essential keywords and possible interpretations are covered. Once these multiple queries are generated, each query is individually sent to the retriever, fetching relevant documents for each variation. However, this results in multiple sets of retrieved documents, often with overlapping content. Instead of passing all retrieved documents to the LLM, a list of unique documents is created and provided along with the original user question to generate a more comprehensive and precise response.

For example, if the user's original query is "Syllabus of dcn", the generated queries may include

- *What is the syllabus for Data Communication and Networking?*
- *What is the complete syllabus for the subject Data Communication and Networking?*
- *What is the curriculum for Data Communication and Networking?*

This approach is effective for retrieving relevant documents, but the method of identifying unique documents needs improvement. The current implementation uses Python's set data structure to eliminate duplicate entries. However, this approach may inadvertently remove highly relevant documents that appear across multiple query results. Since sets do not maintain ranking or contextual importance, an important document could be mistakenly discarded, reducing the overall accuracy of the retrieved information. Thus, a more refined and efficient document retrieval strategy is required to preserve critical context while ensuring query variations enhance, rather than hinder, the response quality.

RAG Fusion:

In his paper, Zackary Rackauckas formally defines Reciprocal Rank Fusion (RRF) as a reranking algorithm that assigns a reciprocal rank to documents retrieved from multiple sources. These ranks are then combined to generate a final re-ranked list of documents. According to his research, RRF outperforms many other reranking methods.

This technique functions similarly to the multi-query approach up to the point of retrieving documents for each generated query. However, instead of using an inefficient method such as a set data structure to create a unique list of documents, RAG fusion employs a specialized ranking algorithm to re-rank all retrieved documents. This algorithm performs a vector search to find a set of relevant documents, much like a standard RAG approach. However, instead of directly passing these retrieved documents along with the queries to the LLM (as done in multi-query retrieval), the model first applies Reciprocal Rank Fusion (RRF).

Reciprocal rank fusion is a widely used search ranking algorithm that assigns scores to retrieved documents and reranks them accordingly. The RRF score for each document is computed using the following formula:

$$rrfscore = \frac{1}{rank + k}$$

where,

rank refers to the document's position in the relevance-sorted list (descending order).

k is a smoothing factor that determines the weight assigned to lower-ranked documents (commonly set to 60, though 1 is used in the example given here for simplification).

When the retriever fetches documents for each sub-query, they are ranked based on their relevance or similarity to the query. The first document in the list is the most relevant, followed by the next most relevant, and so on—making the last document the least relevant. To illustrate how RRF scores are computed, consider the following example:

Example: Computing RRF scores

User question: "Tell me about MEMs microphones"

Generated sub-queries:

1: "What are MEMs microphones and how do they work?"

2: "What are the advantages of using MEMs microphones?"

……

n. [Additional sub-queries]

Retrieved documents and RRF computation:

Sub query 1: Retrieved documents, docs = [docA *(rank 1)*, docB *(rank 2)*….]

$$docA = \frac{1}{1+1} = 0.5$$
$$docB = \frac{1}{1+2} = 0.33$$

Sub query 2: Retrieved documents, docs = [docB *(rank 1)*, docC *(rank 2)*….]

$$docB = 0.33 + \frac{1}{1+1} = 0.33 + 0.5 = 0.83$$
$$docC = \frac{1}{1+2} = 0.33$$

In this example, n sub-queries are generated for the user question, and relevant documents are retrieved for each. Their RRF scores are calculated based on their ranks in the respective retrieval lists. One key observation is that docB appears in multiple retrieval lists. Therefore, when computing its final RRF score, its previous score is added to the newly computed score, making docB the most relevant document for answering the user's question.

After computing the RRF scores for all retrieved documents, they are merged and sorted in descending order based on their scores, ensuring the most relevant document appears first. The model then sends the reranked results, along with the generated sub-queries and the original query, to the LLM to generate a final response.

This re-ranking step is very convenient in cases when we are getting documents from multiple vector stores, or when we want retrieval from a large number of differently worded questions, or in cases where we only want to retrieve the top-k most relevant documents only.

4.2.3 Prompt Engineering

Prompt engineering is the practice of designing effective inputs or instructions for large language models (LLMs) to guide their responses, ensuring accuracy and relevance. While the term "prompt" may seem straightforward and often overlooked due to its simplicity, it plays a crucial role in enhancing the accuracy of a RAG system's outputs. Given that this chatbot is a relatively simple RAG-based application, I sought efficient yet uncomplicated techniques to improve its performance without introducing unnecessary complexity.

A study by Siran Li et al. highlights prompting as the second most important factor in optimizing RAG systems, emphasizing the significance of carefully defining instructions for the model. Keeping this in mind, I explored several widely used prompt engineering techniques, which include:

In-Context Learning (ICL): As the name suggests, ICL enables the model to learn meaning from the context provided within the prompt. By including relevant examples, the model understands patterns and generates appropriate responses. Based on the number of examples used, ICL can be categorized into:

- Zero-shot learning (no examples)
- One-shot learning (a single example)
- Few-shot learning (multiple examples)

In this study, I have implemented all three variations of ICL to guide the LLM and analyzed their responses.

Chain-of-Thought (CoT): This technique instructs the LLM to break down complex problems into smaller, sequential steps, allowing it to approach each step with logical reasoning. By demonstrating step-by-step problem-solving examples, the model is encouraged to think in a structured manner and generate more accurate answers.

Step-by-step reasoning (SSR) enables the model to arrive at an answer through a series of intermediate steps, making it particularly effective for multi-step reasoning tasks such as mathematical problem-solving, logical analysis, and decision-making processes.

Tree of Thought (ToT): The ToT technique structures multiple potential thought processes in a tree-like format, enabling the model to explore different solutions for complex problems. This method is especially useful when multiple reasoning paths need to be considered before arriving at an optimal solution.

Since this chatbot is relatively simple, In-Context Learning (ICL) has been chosen as the primary prompting technique. Three different versions of the chatbot were developed, each implementing a different ICL variation, combined with query rewriting and RAG fusion. The impact of these techniques on response quality and accuracy has been demonstrated in a comparative table. Due to token limitations in the Gemini API, which powers the model, only 15 example queries were tested to showcase the significant differences between each prompting approach.

CHAPTER 5

DISSERTATION DEMONSTRATION

5.1 Algorithms

For storing environment variables:

A .env file is made to store all the environment variables that are to be used throughout the application. This file has API key for Google Gemini model, the LLM used to power the chatbot, Langchain API key which is used with LangSmith to allow tracing of the chatbot. And this file looks something like this:

```
LANGCHAIN_TRACING_V2=false

LANGCHAIN_ENDPOINT="https://api.smith.langchain.com"

LANGCHAIN_API_KEY="lsv2_pt_d0fee456ce2049638355ded8c203cc76_065120
4b6a"

LANGCHAIN_PROJECT="advanced-rag"

GOOGLE_API_KEY="AIzaSyHgtT3NQ1CmcWsTNjR6f8ho_5QaVuf1sLQ"
```

Important dependencies:

Before getting to start with developing the chatbot, the mentioned list of libraries and packages have been installed. The list of these packages comes under a file called requirements.txt:

```
langchain_core

python-dotenv

streamlit

langchain_community

pypdf

google-generativeai
```

```
chromadb
langchain_chroma
langchain_google_genai
```

For converting PDFs into text documents (.txt files):

```python
from pypdf import PdfReader
import os
pdf_dir = "./pdf_docs"
output_dir = "./new_docs"
if not os.path.exists(output_dir):
 os.makedirs(output_dir)
for filename in os.listdir(pdf_dir):
 full_name = os.path.join(pdf_dir, filename)
 reader = PdfReader(full_name)
 # opening text file with same name but idfferent extension
 text_filename = filename.replace(".pdf", ".txt")
 full_textfilename = os.path.join(output_dir, text_filename)
 with open(full_textfilename, "w", encoding="utf-8") as f:
   for page_num, page in enumerate(reader.pages):
     text = page.extract_text()
     if text:
       f.write(text)
       f.write("\n")
```

For removing unnecessary spacing from “syllabus” documents:

After converting PDFs to text files, there found a lot of unnecessary spacing. So rather than manually removing the spaces, below given python code has been used to remove spaces automatically.

```
import re
trigger_string = "Course Code Course Title L T P C"
with open("./new_docs/bsc.txt", "r", encoding="utf-8") as file:
    content = file.read()
cleaned_content = " ".join(content.split()).strip()
pattern = re.compile(rf"(\b{re.escape(trigger_string)}\b)")
formatted_content = re.sub(pattern, r"\n\n\1", cleaned_content)
output_file = "./done_docs/bsc.txt"
with open(output_file, "w", encoding="utf-8") as file:
    file.write(formatted_content)
print(f" Text formatted successfully! Saved to '{output_file}'")
```

For converting text files into equivalent embeddings:

Here for converting text into embeddings, GoogleGeminiAIEmbedding has been used. And for storing these embeddings ChromaDB vector store is used. The embeddings are stored locally.

```
from langchain_community.document_loaders import TextLoader
from langchain.text_splitter import CharacterTextSplitter
from dotenv import load_dotenv
from langchain_google_genai import GoogleGenerativeAIEmbeddings
from langchain_community.vectorstores import Chroma
import os
```

```
load_dotenv()

embedding_model =
GoogleGenerativeAIEmbeddings(model="models/embedding-001")

documents = []

dir_path = "./alldocs"

for file in os.listdir(dir_path):

 if file.endswith(".txt"):

   complete_file_path = os.path.join(dir_path, file)

   raw_doc_list = TextLoader(complete_file_path).load()

   documents.extend(raw_doc_list)

text_splitter = CharacterTextSplitter(chunk_size=1024,
chunk_overlap=500)

text_chunks = text_splitter.split_documents(documents)

vectordb = Chroma.from_documents(

   documents=text_chunks,

   embedding=embedding_model,

   persist_directory="./vectordb"

)

print("Embedding complete !!")
```

Code of Zero-shot prompting technique with query rewriting and rag fusion:

```
import os

from dotenv import load_dotenv

from langchain_community.vectorstores import Chroma

from langchain_google_genai import ChatGoogleGenerativeAI,
GoogleGenerativeAIEmbeddings

from langchain_core.prompts import ChatPromptTemplate,
```

```
MessagesPlaceholder, PromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_core.load import dumps, loads
from langchain_core.messages import HumanMessage, AIMessage
from operator import itemgetter
load_dotenv()
os.environ["LANGCHAIN_API_KEY"] = os.getenv("LANGCHAIN_API_KEY")
os.environ["LANGCHAIN_TRACING_V2"]="true"
persist_directory = "../vectordb"
embeddings                                                         =
GoogleGenerativeAIEmbeddings(model="models/embedding-001")
vectorstore = Chroma(
 persist_directory=persist_directory,
 embedding_function = embeddings
)
retriever = vectorstore.as_retriever()
model = ChatGoogleGenerativeAI(model="gemini-1.5-flash")
# prompting template for generating multi-queries
template = """You are an AI language model assistant. Your task is
to generate five different versions of the given user question to
retrieve relevant documents from the vector database. By
generating multiple perspectives on the user question, your goal
is to help the user overcome some of the limitations of the
distance-based similarity search. Provide these alternative
questions separated by newlines. Original question: {question}"""

multiquery_prompt = PromptTemplate(
 template=template,
 input_variables=['question']
)
```

```
multiquery_chain = (
 multiquery_prompt
 | model
 | StrOutputParser()
 | (lambda x: [q.strip() for q in x.split('\n') if q.strip()])
)
def reciprocal_rank_fusion(results: list[list], k = 60):
 # dict to store new rrf scores for each document
 rrf_scores = {}
 for docs in results:
    # here "rank" means the index of the doc, they are arranged in the order of most relevent - to least
   for rank, doc in enumerate(docs):
     doc_str = dumps(doc)
     if doc_str not in rrf_scores:
       rrf_scores[doc_str] = 0
         # if the doc is not available in the list, add it with initial score of 0
     # get the previous rrf_score of the doc, if any
     prev_score = rrf_scores[doc_str]
     # update the new scores
     rrf_scores[doc_str] = 1 / (rank + k)
 # ------------- arranging the scores in descending order
 reranked_scores = [
   (loads(doc), score)
     for doc, score in sorted(rrf_scores.items(), key=lambda x: x[1], reverse=True)
 ]
 return reranked_scores
```

```
chat_history = []

while True:

 user_input = input('You: ')

 if user_input == 'exit':

break

  chat_history.append(HumanMessage(content=user_input))

 # get docs of each question

    doc_chain    =    multiquery_chain    |    retriever.map()    |
reciprocal_rank_fusion

 # doc_chain = [docs]

# -------------- zero shot prompting

   rag_template   =   """Answer   the   following   question   based   on   this
context:{context}. Question: {question}"""

   # Correct implementation of rag_prompt

 rag_prompt = ChatPromptTemplate.from_messages([

    ("system", rag_template),

    MessagesPlaceholder(variable_name="chat_history"),

    ("human", "{question}")

 ])

 chain = (

      {'context':  doc_chain,  'question':  itemgetter('question'),
'chat_history': itemgetter('chat_history')}

   | rag_prompt

   | model

   | StrOutputParser()

 )

  result  =  chain.invoke({'question':  user_input,  'chat_history':
chat_history})

 chat_history.append(AIMessage(content=result))
```

```
 print(f'AI: {result}')
print(f'\n\n Chat history: \n {chat_history}')
```

Code of One-shot prompting technique with query rewriting and rag fusion:

Every code is same, except for the final `rag_template` that we provide. In one-shot prompting an example has been added as shown below:

```
rag_template = """You are an AI assistant answering questions based on the provided context. Here is an example of how you should respond:\n

 Context: \nAcademic Calendar(2024 - 2025)Winter Semester

  Academic Calendar for Winter Semester 2024 - 25 (Applicable to the students of all the programmes except BSc. (Agri) & MBA)

 date: 09.10.2024 & 10.10.2024

 day: Wednesday & Thursday

 description: Course wish list registration by students

 date: 14.10.2024 - 29.10.2024

 day: Monday to Tuesday

 description: Course allocation and scheduling by Schools

 date: 07.11.2024

 day: Thursday

 description: Mock - Course registration for Freshers

 date: 09.11.2024

 day: Saturday

 description: Course registration by students

 date: 13.12.2024

 day: Friday

 description: Commencement of Winter Semester 2024-25

 date: 13.12.2024-15.12.2024
```

day: Friday to Sunday

description: Course add/drop option for students

date: 22.12.2024-01.01.2025

day: Sunday to Wednesday

description: Winter Vacation for the students (11 Days)

date: 05.01.2025

day: Sunday

description: Last date for the payment of re-registration fees

date: 14.01.2025

day: Tuesday

description: Pongal (Holiday)

date: 27.01.2025-02.02.2025

day: Monday to Sunday

description: Continuous Assessment Test -1

date: 20.02.2025-23.02.2025

day: Thursday to Sunday

description: Riviera 2025

date: 24.02.2025-26.02.2025

day: Monday to Wednesday

description: Course withdraw option for students

date: 14.03.2025

day: Friday

description: Holi (Holiday)

date: 16.03.2025-22.03.2025

day: Sunday to Saturday

description: Continuous Assessment Test - II

date: 30.03.2025

day: Sunday

description: Telugu New Year's Day

date: 31.03.2025

day: Monday

description: Ramzan (No Instructional Day)

date: 05.04.2025

day: Saturday

description: Last instructional day for laboratory classes

date: 07.04.2025-11.04.2025

day: Monday to Friday

description: Final assessment test for laboratory courses/components

date: 14.04.2025

day: Monday

description: Tamil New Year's Day/ Dr. B. R. Ambedkar Birthday(Holiday)

date: 17.04.2025

day: Thursday

description: Last instructional day for theory classes

date: 18.04.2025

day: Friday

description: Good Friday (No Instructional Day)

date: 21.04.2025

day: Monday

description: Commencement of final assessment test for theorycourses / components

date: 12.05.2025

day: Monday

description: Commencement of Summer Term 2024-25 (Tentative)

```
date: 14.07.2025

day: Monday

description: Commencement of Fall Semester 2025-26 (Tentative)

Question: last working day of winter semester 2025?

Answer: The last intructional day for theory classes is 18.04.2025 (friday) and lab classes as (07.04.2025-11.04.2025).

Do not mention explicitly that you're providing the answer from the given context by saying "The provided text mentions" or "Based on provided text". Just directly give the appropriate answer as if the user thinks that you have this knowledge inbuilt in you.

Now, answer the following question based on this context:

Context: {context}

Question: {question}"""
```

Code of Few-shot prompting technique with query rewriting and rag fusion:

Here also, the base code remains the same, but in few-shot prompting multiple examples has been given in prompt to the LLM, covering different aspects of questions from different PDF documents. In case of few-shot prompting `rag_template` would look something like this:

```
rag_template = """You are an AI assistant answering questions based on the provided context. Here are some examples of how you should respond:\n

    date: 07.11.2024

  day: Thursday

  description: Mock - Course registration for Freshers

  date: 09.11.2024

  day: Saturday

  description: Course registration by students

  date: 13.12.2024

  day: Friday
```

description: Commencement of Winter Semester 2024-25

date: 13.12.2024-15.12.2024

day: Friday to Sunday

description: Course add/drop option for students

date: 22.12.2024-01.01.2025

day: Sunday to Wednesday

description: Winter Vacation for the students (11 Days)

date: 05.01.2025

day: Sunday

description: Last date for the payment of re-registration fees

date: 14.01.2025

day: Tuesday

description: Pongal (Holiday)

date: 27.01.2025-02.02.2025

day: Monday to Sunday

description: Continuous Assessment Test -1

date: 20.02.2025-23.02.2025

day: Thursday to Sunday

description: Riviera 2025

date: 24.02.2025-26.02.2025

day: Monday to Wednesday

description: Course withdraw option for students

date: 14.03.2025

day: Friday

description: Holi (Holiday)

date: 16.03.2025-22.03.2025

day: Sunday to Saturday

description: Continuous Assessment Test - II

date: 30.03.2025

day: Sunday

description: Telugu New Year's Day

date: 31.03.2025

day: Monday

description: Ramzan (No Instructional Day)

date: 05.04.2025

day: Saturday

description: Last instructional day for laboratory classes

date: 07.04.2025-11.04.2025

day: Monday to Friday

description: Final assessment test for laboratory courses/components

date: 14.04.2025

day: Monday

description: Tamil New Year's Day/ Dr. B. R. Ambedkar Birthday(Holiday)

date: 17.04.2025

day: Thursday

description: Last instructional day for theory classes

date: 18.04.2025

day: Friday

description: Good Friday (No Instructional Day)

date: 21.04.2025

day: Monday

description: Commencement of final assessment test for theorycourses / components

date: 12.05.2025

day: Monday

description: Commencement of Summer Term 2024-25 (Tentative)

date: 14.07.2025

day: Monday

description: Commencement of Fall Semester 2025-26 (Tentative)

Question: last working day of winter semester 2025?

Answer: The last intructional day for theory classes is 18.04.2025 (friday) and lab classes as (07.04.2025-11.04.2025).\n\n

Example 2: Syllabus related queries. Show answer in proper format.\n

Context: \n

Syllabus Advanced Database Management Systems(5year_cse)

Course code: CSI2004

Course title:Advanced Database Management Systems

Module:1 Database Design Techniques 5 hours Review of DBMS Techniques – EER – Physical database design and tuning – Advanced transaction processing and Query processing Module:2 Parallel Databases 6 hours Architecture, Data partitioning strategy, Interquery and Intraquery Parallelism -Parallel query optimization Module:3 Distributed Databases 7 hours Structure of distributed database, Advantages, Functions, Distributed database architecture, Allocation, Fragmentation, Replication, Distributed query processing, Distributed transaction processing, Concurrency control and Recovery in distributed database systems. Module:4 Multimedia and Spatial Databases 7 hours Multimedia sources, issues, Multimedia database applications Multimedia database queries -LOB in SQL. Spatial databases -Type of spatial data – Indexing in spatial databases. Module:5 Mobile and Cloud Databases 8 hours 1. Wireless network communication, Location and handoff management, Data processing and mobility, Transaction management in mobile database systems, Database options in the cloud , Changing role of the DBA in the cloud , Moving your databases to the cloud Module:6 Emerging Database Technologies 5 hours Active database – Detective database - Object database - Temporal database - Streaming databases Module:7 Database Security 5 hours Introduction to Database Security Issues –Security Models – Different Threats to databases – Counter measures to deal with these problems Module:8 Recent Trends 2 hours Total Lecture hours: 45 hours Text Book(s) 1. Raghu Ramakrishnan , Database Management Systems , ,4th edition, Mcgraw -Hill,2015 2. Abraham Silberschatz, Henry F. Korth, S. Sudharshan, "Database System Concepts", Seventh Edition, Tata McGraw Hill, 2019. Reference Books 1. RamezElmasri, Shamkant B. Navathe, "Fundamentals of Database Systems", Seventh Edition, Pearson Education, 2016. 2. Vlad Vlasceanu, Wendy A. Neu, Andy Oram, Sam Alapati, "An Introduction to Cloud Databases", O'Reilly Media, Inc. 2019 3. S.K.Singh, Database Systems: Concepts, Design & Applications, 2nd Edition, Pearson education, 2011 Mode of Evaluation: CAT/ Digital Assignments/ Quiz/ FAT/ Project.

Syllabus Principles of Compiler Design(5year_cse)

Course code: CSI2004

Course title: Principles of Compiler Design

Module:1 Introduction to Compilation and Lexcial Analysis 7 hours Introduction to programming language translators -Structure and phases of a compiler -Design issues - Patterns - lexemes -Tokens -Attributes -Specification of Tokens - Extended Regular expression, Regular expression to Deterministic Finite Automata (Direct method). Module:2 Syntax Analysis –Top Down 5 hours Role of parser - Parse Tree - Elimination of ambiguity - Top down parsing - Recursive Descent parsing - Non Recursive Descent parsing - Predictive Parsing - LL(1) grammars. Module:3 Syntax Analysis –Bottom Up 7 hours Shift Reduce Parsers - Operator Precedence Parsing ,LR parsers: -Construction of SLR parser tables and parsing , CLR parsing -LALR parsing Module:4 Semantics Analysis 6 hours Syntax Directed Definition – Evaluation Order - Applications of Syntax Directed Translation - Syntax Directed Translation Schemes - Implementation of L attributed Syntax Directed Definition. Module:5 Intermediate Code Generation 7 hours

Variants of syntax trees - Three address code - Types – Declarations - Procedures - Assignment Statements - Translation of Expressions - Control Flow - Back Patching - Switch Case Statements. Module:6 Code Optimization 6 hours Loop optimizations - Principal sources of optimization -Introduction to Data Flow Analysis - Basic Blocks - The DAG Representation of Basic Blocks -Loops in Flow Graphs. Module:7 Code Generation & Other Translations Issues 5 hours Issues in the design of a code generator - Target Machine - Next -Use Information - Optimization of basic blocks - Peephole Optimization - Register Allocation and Assignment. Module:8 Recent Trends 2 hours Total Lecture hours: 45 hours Text Book(s) A. V. Aho, Monica S. Lam, Ravi Sethi and Jeffrey D. Ullman, Compilers: Principles, Techniques, & Tools, Second Edition, , Pearson Education, 2007 K. D. Cooper and L. Torczon, Engineering a Compiler, 2nd e dition. Morgan Kaufmann,2011. Reference Books Andrew A.Appel , Modern Compiler Implementation in Java, 2nd edition, Cambridge University Press;, 2002. Allen Holub, Compiler Design in C, Prentice Hall,1990. Torbengidius Mogensen, "Basics of Compiler Design", Springer, 2011. Mode of Evaluation: CAT / Assignment / Quiz / FAT / Project / Seminar

Question: syllabus of Principles of Compiler Design in mtech cse.

Answer: Syllabus of Principles of Compiler Design (Mtech CSE)

Course code: CSI2004

Course title: Principles of Compiler Design

Module:1 Introduction to Compilation and Lexcial Analysis 7 hours Introduction to programming language translators -Structure and phases of a compiler -Design issues - Patterns - lexemes -Tokens -Attributes -Specification of Tokens - Extended Regular expression, Regular expression to Deterministic Finite Automata (Direct method). Module:2 Syntax Analysis –Top Down 5 hours Role of parser - Parse Tree - Elimination of ambiguity - Top down parsing - Recursive Descent parsing - Non Recursive Descent parsing - Predictive Parsing - LL(1) grammars. Module:3 Syntax Analysis –Bottom Up 7 hours Shift Reduce Parsers - Operator Precedence Parsing ,LR parsers: -Construction of SLR parser tables and parsing , CLR parsing -LALR parsing Module:4 Semantics Analysis 6 hours Syntax Directed Definition – Evaluation Order - Applications of Syntax Directed Translation - Syntax Directed Translation Schemes - Implementation of L attributed Syntax Directed Definition. Module:5 Intermediate Code Generation 7 hours Variants of syntax trees - Three address code - Types – Declarations - Procedures - Assignment Statements - Translation of Expressions - Control Flow - Back Patching - Switch Case Statements. Module:6 Code Optimization 6 hours Loop optimizations - Principal sources of optimization -Introduction to Data Flow Analysis - Basic Blocks - The DAG Representation of Basic Blocks -Loops in Flow Graphs. Module:7 Code Generation & Other Translations Issues 5 hours Issues in the design of a code generator - Target Machine - Next -Use Information - Optimization of basic blocks - Peephole Optimization - Register Allocation and Assignment. Module:8 Recent Trends 2 hours Total Lecture hours: 45 hours Text Book(s) A. V. Aho, Monica S. Lam, Ravi Sethi and Jeffrey D. Ullman, Compilers: Principles, Techniques, & Tools, Second Edition, , Pearson Education, 2007 K. D. Cooper and L. Torczon, Engineering a Compiler, 2nd e dition. Morgan Kaufmann,2011. Reference Books Andrew A.Appel , Modern Compiler Implementation in Java, 2nd edition, Cambridge University Press;, 2002. Allen Holub, Compiler Design in C, Prentice Hall,1990. Torbengidius Mogensen, "Basics of Compiler Design", Springer, 2011. Mode of Evaluation: CAT / Assignment / Quiz / FAT / Project / Seminar. \n\n

Example 3: Faculty related queries

Context:\n

10873

Dr. Senthil Kumar P

Machine Learning, Big Data Analytics, Deep Learning

9994626135

psenthilkumar@vit.ac.in

TUESDAY, 11:30 AM - 12:30 PM, THURSDAY, 03:00 PM - 04:00 PM

SJT-210-A13

11627

Dr. Senthil Murugan B

Cloud Computing, Big Data Analytics, Future Internet (Internet of Things, Internet of Services , Internet of Content), AI & Machine\nLearning, Data Engineering

9047151090

senthilmurugan.b@vit.ac.in

WEDNESDAY, 10:00 AM - 11:00 AM, FRIDAY, 12:00 PM - 01:00 PM

SJT-411-A20

12340

Dr. Shynu P G

Cloud computing, Machine Learning, Big Data Analytics, IoT, Block Chain Technology

9840800396

pgshynu@vit.ac.in

No information

SJT-116-A02

12388

Dr. Srinivas Koppu

Blockchain, IoT, Federated Learning, AI, Data Analytics

7667163460

srinukoppu@vit.ac.in

THURSDAY, 10:00 AM - 11:00 AM, WEDNESDAY, 11:15 AM - 12:45 PM, TUESDAY, 11:00 AM - 12:15 PM

SJT-111-A12

Question: information of shynu sir.

Answer: \n

Emp ID: 12340

Faculty Name: Dr. Shynu P G

Area of specialization: Cloud computing, Machine Learning, Big Data Analytics, IoT, Block Chain Technology

Mobile No.: 9840800396

E-mail ID: pgshynu@vit.ac.in

Open Hours: No information provided

Cabin Number: SJT-116-A02 \n\n

Example 4: Answering faculty related information. In cases where there are more than one faculty or professor with the same name, get the required information of all the faculty having same name if full name is not mentioned properly. Ask for follow-up question if required but also provide details of all faculty or professor having same name. \n

Context:\n

14795

Dr. Ramkumar T

Big Data Analytics, Data Mining

9442421674

ramkumar.thirunavukarasu@vit.ac.in

MONDAY, 09:30 AM - 11:30 AM, TUESDAY, 08:30 AM - 09:45 AM, WEDNESDAY, 10:00 AM - 12:00 PM, THURSDAY, 10:00 AM - 12:00 PM, FRIDAY, 12:00 PM - 01:00 PM

SJT-210-A39

11160

Dr. Senthil Kumaran U

Wireless and Mobile, Data Mining, Software Engineering, Cloud Computing

9994863891

usenthilkumaran@vit.ac.in

No information

SJT-313-A01

12295

Dr. Senthil kumar M

Machine Learning, Deep Learning, Big Data, Security

9994267718

senthilkumar.mohan@vit.ac.in

WEDNESDAY, 10:30 AM - 12:30 AM, FRIDAY, 10:30 AM - 12:30 PM

SJT-116-A11

10873

Dr. Senthil Kumar P

Machine Learning, Big Data Analytics, Deep Learning

9994626135

psenthilkumar@vit.ac.in

TUESDAY, 11:30 AM - 12:30 PM, THURSDAY, 03:00 PM - 04:00 PM

SJT-210-A13

11627

Dr. Senthil Murugan B

Cloud Computing, Big Data Analytics, Future Internet (Internet of Things, Internet of Services , Internet of Content), AI & Machine\nLearning, Data Engineering

9047151090

senthilmurugan.b@vit.ac.in

WEDNESDAY, 10:00 AM - 11:00 AM, FRIDAY, 12:00 PM - 01:00 PM

SJT-411-A20

11599

Dr. Seetha R

Mobile ad hoc network, Cryptography, Blockchain, IoT

9486341417

rseetha@vit.ac.in

MONDAY, 12:00 PM - 01:00 PM, THURSDAY, 04:00 PM - 05:00 PM

SJT-213-A32

10364

Dr. Senthilkumar N.C

Data Mining, Machine Learning, Deep Learning, Big Data Analytics, Sentiment Analysis. Opinion mining, Text mining, IoT, Image\nProcessing

9486845866

ncsenthilkumar@vit.ac.in

No information

SJT-313-A35

12374

Dr. Ramya.G

Formal languages and automata theory Machine learning, Artificial intelligence

9597218281

ramya.g@vit.ac.in

THURSDAY, 01:00 PM - 05:00 PM\nMONDAY, 10:00 AM - 01:00 PM

SJT-116-A38

11581

Dr. Senthilkumar N

Semantic Web, Information Retrieval and Natural Language Processing

9943541777

senthilkumar.n@vit.ac.in

MONDAY, 11:00 AM - 12:00 PM\nTHURSDAY, 02:00 PM - 03:00 PM

SJT-310-A22

19610

Dr. Senthilkumar T

Image Processing, Networking, Wireless & Ad hOC Networks

9442254807

senthilkumar.t@vit.ac.in

No information

PRP BLOCK 218 L

Question: cabin number and open hours of sentil sir.

Answer:\n

There are more than one sentil sir. The cabin number and open hours of all are provided below.

Faculty Name: Dr. Senthilkumar T

Open Hours: No information provided

Cabin Number: PRP BLOCK 218 L

Faculty Name: Dr. Senthilkumar N

Open Hours: MONDAY, 11:00 AM - 12:00 PM\nTHURSDAY, 02:00 PM - 03:00 PM

Cabin Number: SJT-310-A22

Faculty Name: Dr. Senthilkumar N.C

Open Hours: No information provided

Cabin Number: SJT-313-A35

Faculty Name: Dr. Senthil Murugan B

Open Hours: WEDNESDAY, 10:00 AM - 11:00 AM, FRIDAY, 12:00 PM - 01:00 PM

Cabin Number: SJT-411-A20

Faculty Name: Dr. Senthilkumar N.C

Open Hours: No information provided

Cabin Number: SJT-313-A35

Faculty Name: Dr. Senthil Murugan B

Open Hours: WEDNESDAY, 10:00 AM - 11:00 AM, FRIDAY, 12:00 PM - 01:00 PM

Cabin Number: SJT-411-A20

Faculty Name: Dr. Senthil Kumaran U

Open Hours: No information

Cabin Number:SJT-313-A01

Faculty Name: Dr. Senthil kumar M

Open Hours:WEDNESDAY, 10:30 AM - 12:30 AM, FRIDAY, 10:30 AM - 12:30 PM

Cabin Number:SJT-116-A11

Faculty Name: Dr. Senthil Kumar P

Open Hours:TUESDAY, 11:30 AM - 12:30 PM, THURSDAY, 03:00 PM - 04:00 PM

Cabin Number:SJT-210-A13\n\n

Do not mention explicitly that you're providing the answer from the given context by saying "The provided text mentions" or "Based on provided text". Just directly give the appropriate answer as if the user thinks that you have this knowledge inbuilt in you. If you are unable to frame any answer, do not make it up, and simply say "Unable to answer this question at this moment." If someone asks who are you, answer response that you're assistant designed to help students of VIT. If someone asks who is your developer or who built you, answer a student from VIT only. Give your responses in proper format, and use pointers wherever needed.

```
Now, answer the following question based on this context:

Context: {context}

Question: {question}"""
```

Final Code for Enhanced RAG chatbot:

Streamlit has been used to build an engaging UI of the chatbot. When dealing with streamlit, there is a slight different way to address how history is being stored here. The code of everything combined is shown below:

```
import os

from dotenv import load_dotenv

from langchain_community.vectorstores import Chroma

from langchain_google_genai import ChatGoogleGenerativeAI, GoogleGenerativeAIEmbeddings

from langchain_core.prompts import ChatPromptTemplate, MessagesPlaceholder, PromptTemplate

from langchain_core.output_parsers import StrOutputParser

from langchain_core.load import dumps, loads

from operator import itemgetter

import streamlit as st

load_dotenv()

os.environ["LANGCHAIN_API_KEY"] = os.getenv("LANGCHAIN_API_KEY")

os.environ["LANGCHAIN_TRACING_V2"]="true"

persist_directory = "../vectordb"

embeddings = GoogleGenerativeAIEmbeddings(model="models/embedding-001")

vectorstore = Chroma(

 persist_directory=persist_directory,

 embedding_function = embeddings
```

```
)

retriever = vectorstore.as_retriever()

model = ChatGoogleGenerativeAI(model="gemini-1.5-flash")



template = """You are an AI language model assistant. Your task is
to generate five different versions of the given user question to
retrieve relevant documents from the vector database. By
generating multiple perspectives on the user question, your goal
is to help the user overcome some of the limitations of the
distance-based similarity search. Provide these alternative
questions separated by newlines. Original question: {question}"""

# Correct implementation of multiquery_prompt

multiquery_prompt = PromptTemplate(

 template=template,

 input_variables=['question']

)

multiquery_chain = (

 multiquery_prompt

 | model

 | StrOutputParser()

 | (lambda x: [q.strip() for q in x.split('\n') if q.strip()])

)

def reciprocal_rank_fusion(results: list[list], k = 60):

 rrf_scores = {}

 for docs in results:

   for rank, doc in enumerate(docs):

     doc_str = dumps(doc)

     if doc_str not in rrf_scores:

       rrf_scores[doc_str] = 0
```

```
      prev_score = rrf_scores[doc_str]
      rrf_scores[doc_str] = 1 / (rank + k)
 reranked_scores = [
   (loads(doc), score)
      for doc, score in sorted(rrf_scores.items(), key=lambda x: x[1], reverse=True)
 ]
 return reranked_scores



st.header('VIT Chatbot')
st.title('I am here to help you. Ask me anything.')
# defining history if its not already there
if "chat_history" not in st.session_state:
 st.session_state.chat_history = []
# display all messages on the screen
for message in st.session_state.chat_history:
 with st.chat_message(message['role']):
   st.markdown(message['content'])
user_input = st.chat_input('Ask me anything.')
if user_input:
 # display it on screen
 with st.chat_message('user'):
   st.markdown(user_input)
 # append user input to history
  st.session_state.chat_history.append({'role': 'user', 'content': user_input})
 # display chatbot output
```

```
 with st.chat_message('assistant'):

        doc_chain   =   multiquery_chain   |   retriever.map()   |
reciprocal_rank_fusion

   # -------------- few shot prompting

    rag_template = """You are an AI assistant answering questions
based on the provided context. Here are some examples of how you
should respond:\n

<examples in few-shot prompting has been trimmed down here to
increase readability of the code>

Do not mention explicitly that you're providing the answer from
the given context by saying "The provided text mentions" or "Based
on provided text". Just directly give the appropriate answer as if
the user thinks that you have this knowledge inbuilt in you. If
you are unable to frame any answer, do not make it up, and simply
say "Unable to answer this question at this moment." If someone
asks who are you, answer response that you're assistant designed
to help students of VIT. If someone asks who is your developer or
who built you, answer a student from VIT only. Give your responses
in proper format, and use pointers wherever needed. Now, answer
the following question based on this context:

Context: {context}

Question: {question}"""

# Correct implementation of rag_prompt

   rag_prompt = ChatPromptTemplate.from_messages([

     ("system", rag_template),

     MessagesPlaceholder(variable_name="chat_history"),

     ("human", "{question}")

   ])

   chain = (

       {'context': doc_chain, 'question': itemgetter('question'),
'chat_history': itemgetter('chat_history')}

     | rag_prompt

     | model

     | StrOutputParser()
```

```
    )

result   =   chain.invoke({'question':   user_input,   'chat_history':
st.session_state.chat_history})

    st.markdown(result)

        st.session_state.chat_history.append({'role':   'assistant',
'content': result})
```

5.2. Snapshots

Tracing snapshots of enhanced chatbot using LangSmith:

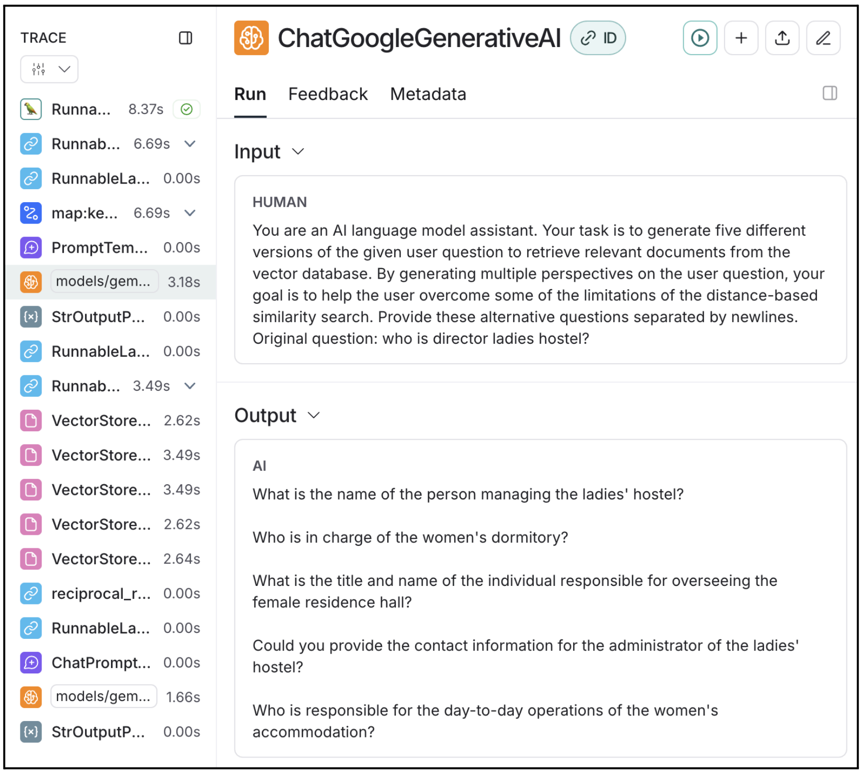

Fig 5.1. LangSmith tracer snapshot showing how multi-queries are being generated

Snapshots of Working of the chatbot:

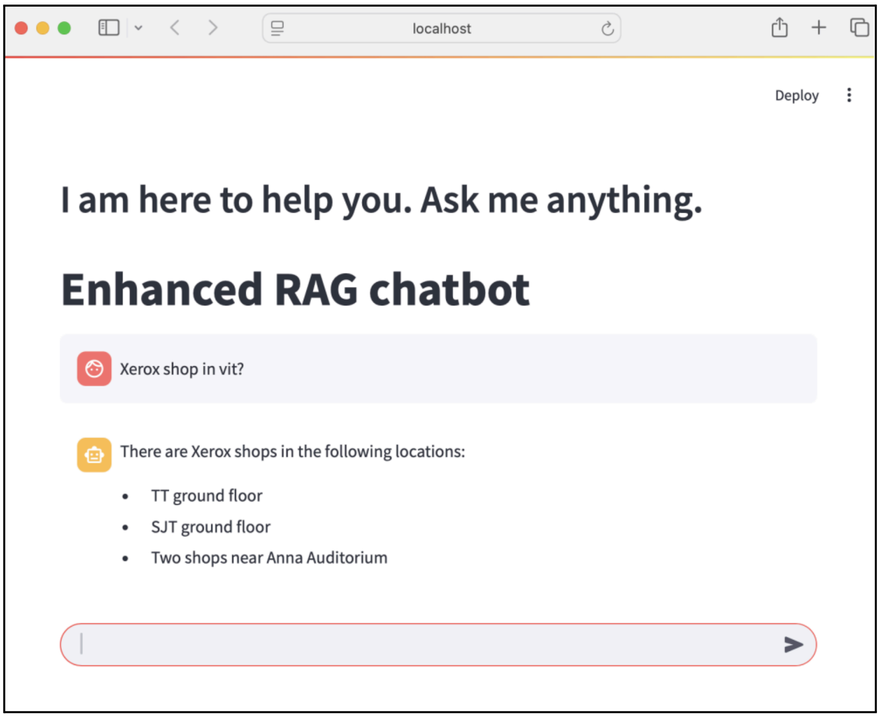

Fig 5.2. Working of enchanced chatbot showing list of xerox shops

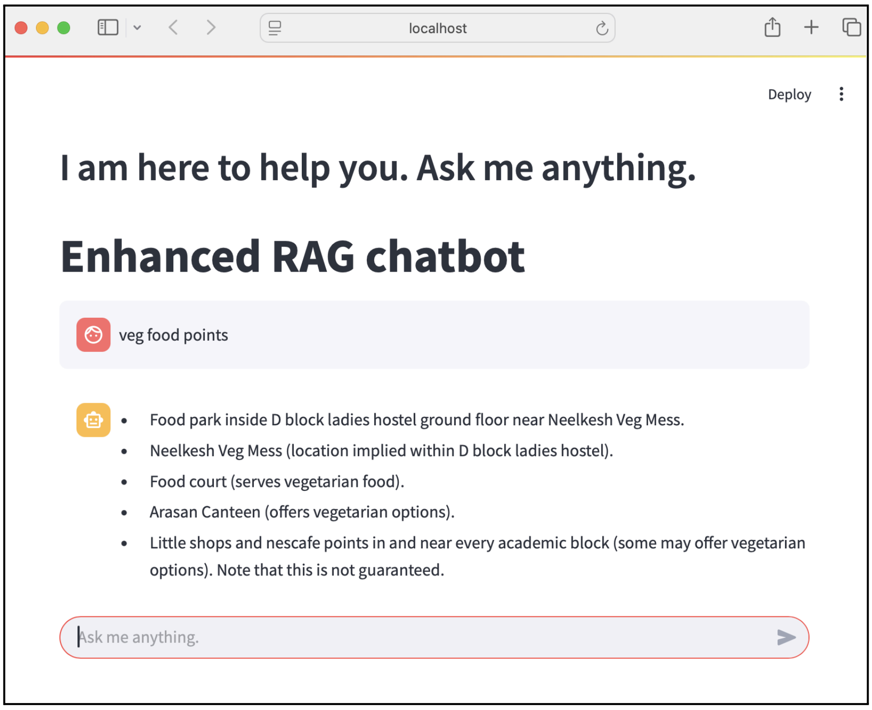

Fig 5.3. Chatbot showing list of food points in VIT

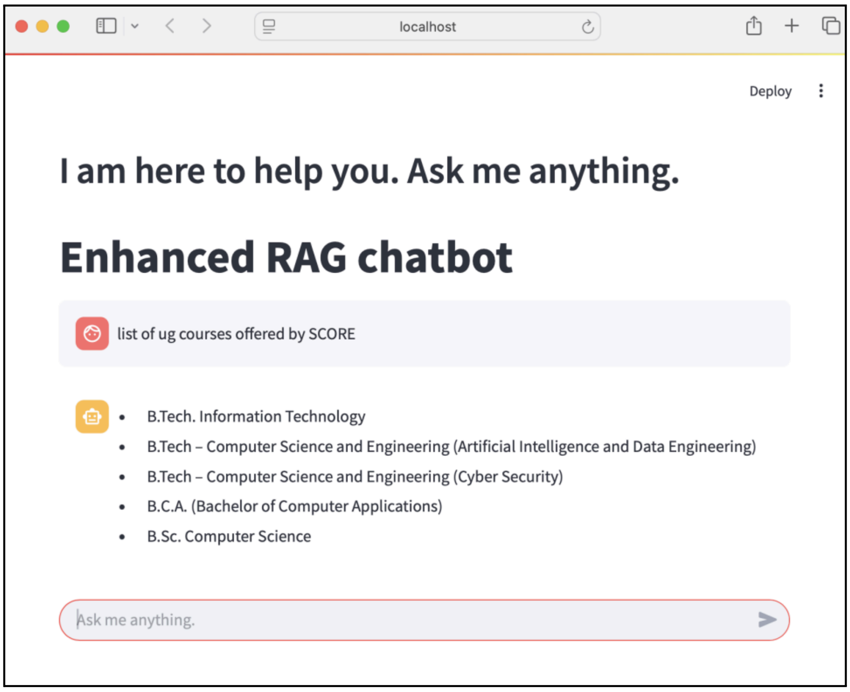

Fig 5.4. Chatbot answering the query of showing UG course in SCORE

CHAPTER 6

RESULT AND DISCUSSION

To improve the accuracy of the naive implementation of the RAG chatbot, an enhanced version has been developed. The RAG pipeline offers a variety of advanced techniques at each stage; however, selecting the most suitable approach depends on the specific use case. Since this chatbot does not require complex reasoning and primarily retrieves answers from structured documents, the enhancements focus on fundamental yet effective techniques that can be easily implemented in similar lightweight RAG-based applications. The improvements in this version include query rewriting, which generates a list of reformulated queries based on the user's input, RAG fusion, which effectively ranks and retrieves the most relevant documents, and prompting techniques—with few-shot prompting used in the final implementation—to refine the chatbot’s responses. Additionally, the dataset has been expanded to include hostel information, syllabus details for courses under SCORE, and general information about shops and food points at VIT, further enhancing the chatbot’s ability to address a wider range of student queries.

Table 6.1: Table showing the actual query asked and the equivalent multi-queries generated

Query	List of questions generated
Syllabus of penetration testing and vulnerability assessment	Course materials for a Penetration Testing and Vulnerability Assessment class Curriculum or learning objectives for a Penetration Testing and Vulnerability Assessment course Topics covered in a Penetration Testing and Vulnerability Assessment syllabus Readings and assignments for a Penetration Testing and Vulnerability Assessment subject Outline of a Penetration Testing and Vulnerability Assessment training program
Last working day in winter semester	Course materials for a Penetration Testing and Vulnerability Assessment class

	Curriculum or learning objectives for a Penetration Testing and Vulnerability Assessment course Topics covered in a Penetration Testing and Vulnerability Assessment syllabus Readings and assignments for a Penetration Testing and Vulnerability Assessment subject Outline of a Penetration Testing and Vulnerability Assessment training program
Last working day in winter semester	When does the winter semester end? What is the final day of classes in the winter semester? Winter semester: date of last day of instruction? What's the last day of the winter term? Final day of the winter semester for students?
Cabin number of vijay Anand sir.	What is the cabin number for Vijay Anand? Where is Vijay Anand's office located? (Assuming cabin number implies office location) What is the room number assigned to Vijay Anand? Can I get contact information for Vijay Anand, including his office location? Find the office/cabin assignment for an employee named Vijay Anand.
Cabin number of senthil sir	What is Senthil sir's cabin/room number? Where is Senthil sir's office located? (Assuming cabin refers to an office) Can you provide the location details for Senthil sir? (More general, might retrieve more context) What is the room assignment for Senthil sir? (Slightly more formal) Find the office or cabin details for an employee named Senthil. (More specific, targets employee records)
Syllabus of DCN mtech 5 year	Course details for a 5-year MTech CSE program's Data Communication Networks (DCN) syllabus. MTech CSE 5-year program: syllabus for the Data Communications and Networks course. What topics are covered in the Data Communication Networks (DCN) course within the

	5-year MTech Computer Science and Engineering curriculum? Find the syllabus for the Data Communications and Networks subject in a 5-year Master of Technology (MTech) program in Computer Science and Engineering. Retrieve the curriculum for the Data Communication Networks (DCN) class offered as part of a 5-year integrated MTech in Computer Science and Engineering.
How many holidays are there winter semester 2025	What are the dates of all holidays during the winter semester of the academic year 2025? List all winter semester holidays for the 2025 academic year. Provide a count of the official holidays observed during the winter 2025 semester. What is the total number of non-instructional days in the winter semester of 2025? (assuming holidays are non-instructional) How many days are designated as holidays within the timeframe of the winter 2025 semester?
Courses offered by score?	What courses does Score offer? Courses available at Score institution. Score's course catalog. What academic programs are provided by Score? Curriculum offered by Score.
Employee ID of Alagiri?	What is Alagiri's employee ID number? Find the employee ID associated with the name Alagiri. Retrieve the record for Alagiri and return the employee ID field. Which employee ID corresponds to the employee named Alagiri? Show me the employee ID for the individual named Alagiri.
Total Number of boys hostel, name each one of them	List all boys' hostels and their corresponding counts. How many boys' hostels are there, and what are

	their individual names? Provide a complete list of all boys' hostels. What is the total number of boys' hostels, and what is the name of each? Enumerate all boys' hostels and state the total quantity.

Table 6.2: Table showing responses of each prompting technique

Query	Re-ranking + zero-shot prompting	Re-ranking + one-shot prompting	Re-ranking + few-shot prompting	Observations
last working day in winter semester?	Based on the provided text, the last instructional day/last working day for theory classes in the winter semester is **November 22, 2024**.	The last instructional day for theory classes is April 17th, 2025, and the last day for lab classes is April 11th, 2025.	The last instructional day for theory classes is April 17th, 2025 (Thursday), and for lab classes is April 11th, 2025 (Friday).	The instruction has been added in both one-shot and few-shot to not to say anything like "Based on the provided text". Making a feel that the chatbot has its inbuilt knowledge, and its answering on its own.
information about vijay anand sir	I am sorry, but I don't have access to personal information about individuals, including "Vijay Anand sir." My knowledge is based on the text I was provided, and that	Dr. Vijay Anand R is a faculty member whose expertise includes Blockchain, Networking, and Software Testing. His contact information is not available in this	Emp ID: 14081 Faculty Name: Dr. Vijay Anand R Area of specialization: Block Chain, Networking, Software Testing Mobile No.: 9486925104	As we can see, zero-shot is not able to answer the question, that is perfectly answered by few-shot, in a proper format as well, giving a complete information about vijay anand sir.

	text does not contain any information about this person.	context.	E-mail ID: vijayanand.r@vit.ac.in Open Hours: No information provided Cabin Number: SJT-210-A10	
vijay anand sir cabin number	I apologize, but I do not have access to that information. My knowledge comes from a limited dataset, and it does not include the cabin number of a person named Vijay Anand sir.	Dr. Vijay Anand R's cabin number is SJT-210-A10.	SJT-210-A10	This question was asked based on the previous question and thus few-shot and one-shot has answered accordingly.
cabin number of senthil sir	There are multiple faculty members with the name Senthil in the provided text. To answer your question, I need to know which Senthil sir you are referring to. The documents list several faculty with variations of the name "Senthil" and their respective cabin numbers (or lack thereof).	There are multiple faculty members with the name Senthil. To answer your question accurately, I need to know which Senthil sir you are referring to. The provided text mentions several faculty members with names containing "Senthil".	To answer your question accurately, I need the full name of the "Senthil sir" you're inquiring about, as there are multiple faculty members with variations of the name "Senthil" in the provided data.	All three were able to determine that there are more than one faculty with this name, thus asking for clarification from the user.
give me list of all senthil sir	* **Dr. Senthilkumar N:** Room/Building information not available in this text. * **Dr. Senthilkumar T:** PRP BLOCK 218 L * **Dr. Senthilkumar N.C:** SJT-313-A35 * **Dr. Senthil kumar M:** SJT-116-A11 * **Dr. Senthil Kumar P:** SJT-210-A13 * **Dr. Senthil Murugan B:** SJT-411-A20	The provided text mentions the following faculty members with names containing "Senthil": * Dr. Senthilkumar N * Dr. Senthilkumar T * Dr. Senthilkumar N.C * Dr. Senthil kumar M * Dr. Senthil Kumar P * Dr. Senthil Murugan B Please note that this list is based solely on	Here's a list of all the faculty members with names containing "Senthil" from the provided data: * **Dr. Senthil Kumar P:** SJT-210-A13 * **Dr. Senthil Murugan B:** SJT-411-A20 * **Dr. Senthil kumar M:** SJT-116-A11 * **Dr. Senthilkumar N.C:** SJT-313-A35 * **Dr. Senthilkumar N:** SJT-310-A22	The note provided by zero-shot and one-shot shows that there maybe a possibility of this list being incomplete. A note like this one gives a sense that the answers provided by the chatbot may not be accurate.

	Please note that the provided text may not be fully comprehensive. There might be additional faculty members named Senthil not included in these documents.	the provided text. There may be other faculty members with the name Senthil not included in this dataset.	* **Dr. Senthil Kumaran U:** SJT-313-A01 * **Dr. Senthilkumar T:** PRP BLOCK 218 L Please note that there are variations in spelling and the inclusion of middle names/initials.	
syllabus of DCN in mtech cse 5year	Here's the syllabus for Data Communication and Networks (DCN) from the provided text for the 5-year MTech CSE program: **Course Code:** CSI2007 **Course Title:** Data Communication and Networks **Module 1: Basics of Data Communication and Computer Network (5 hours)** Mentioned all modules.	The provided text includes a syllabus for a course titled "Data Communication and Networks" (course code: CSI2007) in the 5-year CSE program. Here's a summary of the modules covered: **Module 1: Basics of Data Communication and Computer Network (5 hours)** Mentioned all modules	I need access to the relevant documents to provide you with the syllabus for Data Communication and Networks (DCN) in the 5-year MTech CSE program. The information was not included in the context you previously provided.	The response we got from few-shot shows that sometimes the LLM fails to generalize certain terms, that results in it not being able to answer such detailed queries.
how many holidays are there in winter semester 2025?	Based on the provided text, there are two holidays listed for the Winter Semester of 2025: 1. Pongal 2. Holi The Winter Vacation is a period of time off, but not explicitly labeled as a holiday in the same way.	Based on the provided calendar, there are four holidays during the Winter Semester 2025: 1. Pongal 2. Holi 3. Telugu New Year's Day 4. Tamil New Year's Day/Dr. B.R. Ambedkar Birthday The Winter Vacation is also listed, but that's not a single holiday, it is an 11-day break.	There are four holidays in the Winter Semester 2025: Pongal, Holi, Tamil New Year's Day/ Dr. B. R. Ambedkar Birthday, and Good Friday. There is also one day described as "Ramzan (No Instructional Day)".	Zero-shot failed to display all the holidays, on the other hand few-shot perfectly answers the query.
give me dates for each of them	Based on the provided text: * **Pongal:**	Here are the dates for the holidays and non-instructional day	* **Pongal:** January 14, 2025 * **Holi:** March	Zero-shot again missed the correct answer. And answers provided by one-shot

	January 14, 2025 * **Holi:** March 14, 2025	during the Winter Semester 2025: * **Pongal:** January 14, 2025 * **Holi:** March 14, 2025 * **Telugu New Year's Day:** March 30, 2025 * **Tamil New Year's Day/Dr. B. R. Ambedkar Birthday:** April 14, 2025 * **Ramzan (No Instructional Day):** March 31, 2025	14, 2025 * **Tamil New Year's Day/ Dr. B. R. Ambedkar Birthday:** April 14, 2025 * **Good Friday:** April 18, 2025 * **Ramzan (No Instructional Day):** March 31, 2025	and few-shot seems abosulelty fine.
alagiri	Dr. Alagiri I is a faculty member whose contact information and office hours are listed in the provided text. Specifically: * **Name:** Dr. Alagiri I * **Specialization:** Wireless Networks, Data Mining * **Phone:** 9025353585 * **Email:** alagiri@vit.ac.in * **Office Hours:** WEDNESDAY, 11:00 AM - 12:00 PM, FRIDAY, 11:30 AM - 12:30 PM * **Office Location:** SJT-210-A09	I need more context or information to understand what you mean by "alagiri". Is this a name, a place, a word in another language, or something else? Please provide more details.	I am sorry, but I do not have any information about Alagiri in the context you provided earlier. To assist you, I need more information about what you would like to know regarding Alagiri.	Surprisingly, the chatbot performed well with zero-shot in this case. This shows that the more examples we provide, the more we are restricting the model to think and process freely.
what is fees for mca?	The fees for the first year of the Master of Computer Applications (MCA) program is ₹140,000. The second year's tuition is ₹137,000.	The fees for the first year of an MCA program is ₹140,000. The second year's tuition is ₹137,000.	The first-year tuition for an MCA is ₹137,000, with a ₹3,000 refundable caution deposit. The total first-year fee is ₹140,000. The second year's tuition is ₹137,000.	answered by all, with few-shot prompting breaking down the fees structure properly.
fees for ladies hostel	The provided text	I cannot answer your	The document	Few-shot prompting worked better than

4 bedded ac room	gives fees in both Indian Rupees (INR) and US Dollars (USD), but doesn't specify a 4-bedded AC room. There are fees listed for 4-bed Non-AC and 4-bed AC rooms, but the pricing varies depending on the mess plan (veg, non-veg, special) and whether it's for undergraduate (UG) or postgraduate (PG) students. To give you an accurate answer, I need more information. Please specify: * **Undergraduate or Postgraduate:** Are you an undergraduate or postgraduate student? * **Mess Plan:** Which mess plan are you interested in (vegetarian, non-vegetarian, or special)?	question. The document provides pricing for ladies' hostel rooms based on room type (1 AC, 2 AC, 4 AC, etc.) and meal plan, but not the number of beds in each room. There is no information specifying the number of beds per room type.	provides pricing for 4 AC rooms in ladies' hostels, but it doesn't specify whether these are 4-bedded rooms. The cost varies depending on whether the student is an undergraduate (UG) or postgraduate (PG) student. * **For UG & PG students:** The fee is ₹1,65,000 (Veg Mess), ₹1,73,700 (Non-Veg Mess), or ₹1,84,500 (Special Mess). This includes a ₹15,000 admission fee and a ₹15,000 refundable caution deposit. There is no information about a 4-bedded AC room specifically. If you can clarify whether you require the fee for UG or PG students, I can provide a more precise answer.	the other two. Even though the response is not upto the mark, but still it is much descriptive and accurate than the other two.
who is dean of score	Dr. Daphne Lopez is the Dean in-charge of the School of Computer Science Engineering and Information Systems (SCORE).	Dr. Daphne Lopez is the Dean in-charge of the School of Computer Science Engineering and Information Systems (SCORE).	Dr. Daphne Lopez is the Dean in-charge of the School of Computer Science Engineering and Information Systems (SCORE).	A clear mention of SCORE department's dean is there in the documents. Thus all three techniques were able to provide correct response.
syllabus of Software Application Architecture	Here's a summary of the Software Application Architecture syllabus based on the provided text: **Course Code:** CSI3024 **Course Title:** Software Application Architecture **Modules:**	**Course Code:** CSI3024 **Course Title:** Software Application Architecture **Module 1: Design Patterns (4 hours)** * Architecture Styles and Patterns * Design Patterns and Principles * Frameworks	Syllabus Software Application Architecture (5year_cse) Course code: CSI3024 Course title: Software Application Architecture Module: 1 Design Patterns (4 hours) Architecture Styles	Responses got from all three techniques are upto the mark.

	* **Module 1: Design Patterns (4 hours):** Architecture Styles and Patterns, Design Patterns and Principles, Frameworks, Architecture, Enterprise Architecture, Various Architecture Design patterns, Patterns History, MVC Design Patterns, Standards, Benefits… (syllabus has been trimmed. The chatbot gave complete syllabus) This syllabus focuses heavily on design patterns, Java, microservices architecture, and API design and development. The course appears to be practical, with hands-on components like developing Spring Boot Microservices applications.	* Architecture * Enterprise Architecture * Various Architecture Design patterns * Patterns History * MVC Design Patterns * Standards * Benefits **Module 2: Java Patterns (7 hours)** * GOF and JEE Blue Print Patterns * Creational, Structural and Behavioral patterns * Modern Java EE Patterns * Core J2EE Patterns **Module 3: Architecture Types & Microservices Architecture (6 hours)** * What are Microservices? * Monolithic vs. Microservices * Microservices Challenges * Application Architecture Patterns * Service Decomposition * Building Microservices applications (complete syllabus) 2. Fowler, M., Patterns of **Mode of Assessment:** Continuous Assessment Test / Assignments / Quiz / FAT / Project / Seminar	and Patterns, Design Patterns and Principles, Frameworks, Architecture, Enterprise Architecture, Various Architecture Design patterns, Patterns History, MVC Design Patterns, Standards, Benefits. Module: 2 Java Patterns (7 hours) GOF and JEE Blue Print Patterns, Creational, Structural and Behavioural patterns, Modern Java EE Patterns, Core J2EE Patterns. Module: 3 Architecture Types & Microservices Architecture (6 hours) What are Microservices, Monolithic Vs Microservices, Microservices Challenges, Application Architecture Patterns, Service Decomposition, Building Microservices application, Module: 4 Microservices Architecture Tools and Technologies (6 hours) Test / Assignments / Quiz / FAT / Project / Seminar	
list of courses offered by score	Based on the provided text, SCORE (School of Computer Science	Based on the provided text, the School of Computer Science Engineering	Based on the provided text, the School of Computer Science Engineering	

	Engineering and Information Systems) offers the following courses: **Undergraduate (UG):** * B.Tech. Information Technology * B.Tech – Computer Science and Engineering (Artificial Intelligence and Data Engineering) (list of all courses offered by SCORE is provided) (response has been trimmed down for readability) Programme The documents also mention a focus on Information Technology, Artificial Intelligence and Data Engineering, Cyber Security, Software Engineering, Computer Science and Computer Applications, but these are areas of specialization within the listed degree programs, not separate courses themselves.	and Information Systems (SCORE) offers the following programs: **Undergraduate (UG):** * B.Tech. Information Technology * B.Tech – Computer Science and Engineering (Artificial Intelligence and Data Engineering) * B.Tech – Computer Science and Engineering (Cyber Security) * B.C.A. (Bachelor of Computer Applications) * B.Sc. Computer Science **Postgraduate (PG):** * M.Tech [Software Engineering] * MCA (Master of Computer Applications) **Other:** * Full-time Ph.D. Programme The documents also mention specific courses within these programs, but a complete list of every single course offered isn't available in the provided text.	and Information Systems (SCORE) offers the following programs in the Vellore campus: **Undergraduate (UG):** * B.Tech. Information Technology * B.Tech – Computer Science and Engineering (Artificial Intelligence and Data Engineering) (response has been trimmed down for increased readability) The documents also mention that SCORE offers a diverse range of programs in Information Technology, Artificial Intelligence and Data Engineering, Cyber Security, Software Engineering, Computer Science and Computer Applications. However, the specific course names beyond those listed above are not detailed in the provided text.	

Even though various commercially available and widely used datasets exist for testing RAG applications, most of them are not tailored to the education domain. Testing the chatbot on a dataset it has not been trained on would be ineffective, as it would not contribute to improving its accuracy or provide meaningful evaluation results. Therefore, using an inappropriate dataset for testing would not be a valid approach. While I acknowledge that evaluating the chatbot with only 15 queries may not provide a comprehensive assessment of its accuracy, this limitation arose due to the token window constraint of the Gemini model used in the chatbot. Generating a larger set of queries exceeded the quota limit, making it infeasible to conduct a more extensive evaluation. As a result, these 15 queries have been used as a metric to assess the chatbot's accuracy within the given constraints.

CHAPTER 7

SUMMARY

This report presents the development of an enhanced version of the previously developed naive RAG chatbot. While the chatbot was initially designed specifically for VIT, it can be easily adapted for any other educational institution, hence the name "Enhanced RAG Chatbot for Educational Institutions." The report provides a detailed analysis of the advanced techniques incorporated to improve the chatbot's accuracy and efficiency. The development process began by analyzing feedback from the initial version, identifying its shortcomings, and selecting advanced techniques to address those issues. Although multiple techniques exist at each stage of the RAG pipeline, the focus was on choosing only those that effectively aligned with the previously encountered challenges while ensuring improvements were made in a simpler and more efficient manner. Experimentation with various prompting techniques reinforced the idea that returning to foundational concepts often yields the best results, as sometimes the simplest and most obvious approach can effectively resolve a problem. Analysis of user queries from the initial prototype revealed that students were not only interested in courses and syllabi but were also seeking general information about food points, hangout spots, shops, departments, hostels, and other campus facilities. To accommodate these needs, additional data was collected and integrated into the system. Since publicly available datasets for evaluating educational chatbots are limited and not domain-specific, a custom evaluation set consisting of 15 carefully curated questions was created. The chatbot's responses were then compared and analyzed in a comparative study, showcasing the accuracy and quality improvements in the enhanced version.

7.1 Conclusion

The development of the Enhanced RAG Chatbot for Educational Institutions demonstrates the impact of incorporating advanced techniques within the RAG pipeline to improve accuracy, efficiency, and contextual relevance in chatbot responses. By analyzing the shortcomings of the initial naive implementation, key areas requiring improvement were identified, leading to the strategic integration of query decomposition, re-ranking using Reciprocal Rank Fusion (RRF), and prompt engineering. These enhancements have significantly refined the chatbot's ability to understand user queries, retrieve highly relevant information, and generate well-structured responses. Furthermore, the addition of domain-specific data covering courses, hostels, campus facilities, and common student queries has made the chatbot more comprehensive and user-centric. The comparative study between the initial and enhanced versions highlights the effectiveness of these improvements, showcasing the chatbot's ability to deliver more accurate and contextually appropriate responses. This project serves as a practical example of how Retrieval-Augmented Generation (RAG) can be optimized for domain-specific applications, particularly in the education sector. The methodologies and insights gained from this study can be extended to other institutions and domains, enabling further automation of repetitive query-handling tasks.

CHAPTER 8

FUTURE WORK

Moving forward, additional refinements such as multi-turn conversations, better personalization, and real-time document updates could further enhance the chatbot's capabilities, making it an even more effective tool for educational institutions. Future improvements will focus on exploring advanced routing and indexing techniques to optimize document retrieval and response generation. Additionally, the integration of AI agents in an agentic AI framework will be investigated to enable more autonomous decision-making and contextual adaptability, further enhancing the chatbot's efficiency and scalability.

CHAPTER 9

REFERENCES

[1] Rackauckas, Z. (2024). Rag-fusion: a new take on retrieval-augmented generation. *arXiv preprint arXiv:2402.03367*.

[2] Li, S., Stenzel, L., Eickhoff, C., & Bahrainian, S. A. (2025). Enhancing Retrieval-Augmented Generation: A Study of Best Practices. *arXiv preprint arXiv:2501.07391*.

[3] Son, M., Won, Y. J., & Lee, S. (2025). Optimizing Large Language Models: A Deep Dive into Effective Prompt Engineering Techniques. *Applied Sciences (2076-3417)*, *15*(3).

[4] Gao, L., Ma, X., Lin, J., & Callan, J. (2023, July). Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 1762-1777).

[5] Maryamah, M., Irfani, M. M., Raharjo, E. B. T., Rahmi, N. A., Ghani, M., & Raharjana, I. K. (2024, February). Chatbots in academia: a retrieval-augmented generation approach for improved efficient information access. In *2024 16th International Conference on Knowledge and Smart Technology (KST)* (pp. 259-264). IEEE.

[6] Patel, S., Patel, J., Shah, D., Goel, P., & Patel, B. (2024, November). A RAG based Personal Placement Assistant System using Large Language Models for Customized Interview Preparation. In *2024 5th International Conference on Data Intelligence and Cognitive Informatics (ICDICI)* (pp. 1468-1475). IEEE.

[7] Alzubi, J. A., Jain, R., Singh, A., Parwekar, P., & Gupta, M. (2023). COBERT: COVID-19 question answering system using BERT. *Arabian journal for science and engineering*, *48*(8), 11003-11013.

[8] Zheng, H. S., Mishra, S., Chen, X., Cheng, H. T., Chi, E. H., Le, Q. V., & Zhou, D. (2023). Take a step back: Evoking reasoning via abstraction in large language models. *arXiv preprint arXiv:2310.06117*.

[9] Wijaya, O. C., & Purwarianti, A. (2024, September). An Interactive Question-Answering System Using Large Language Model and Retrieval-Augmented Generation in an Intelligent Tutoring System on the Programming Domain. In *2024 11th International Conference on Advanced Informatics: Concept, Theory and Application (ICAICTA)* (pp. 1-6). IEEE.

[10] Heredia, J. A., & Barreda, J. G. (2024). An advanced retrieval-augmented generation system for manufacturing quality control. *Available at SSRN 4828863*.

[11] Boroş, T., Chivereanu, R., Dumitrescu, S., & Purcaru, O. (2024, May). Fine-Tuning and Retrieval Augmented Generation for Question Answering Using Affordable Large Language Models. In *Proceedings of the Third Ukrainian Natural Language Processing Workshop (UNLP)@ LREC-COLING 2024* (pp. 75-82).

[12] Singh, D., Tomer, A., Kaushik, T. K., Singh, R., & Kumar, S. (2024, May). The Potential Applications of Artificial Intelligence and Machine Learning in India's Legal System. In *2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE)* (pp. 820-826). IEEE.

[13] Omrani, P., Hosseini, A., Hooshanfar, K., Ebrahimian, Z., Toosi, R., & Akhaee, M. A. (2024, April). Hybrid Retrieval-Augmented Generation Approach for LLMs Query Response Enhancement. In *2024 10th International Conference on Web Research (ICWR)* (pp. 22-26). IEEE.

[14] Niu, J. (2023, August). ML-DPR: A Meta-Learning-Based Model for Domain-Adaptive Dense Passage Retrieval. In *2023 IEEE International Conference on Image Processing and Computer Applications (ICIPCA)* (pp. 985-989). IEEE.

[15] Jaiswal, A., Tiwari, G., Jha, A., Mangulkar, R., & Rani, P. (2024, August). Retrieval Augmented Generation Approach for Multipdf Chatbot using LangChain. In *2024 8th International Conference on Computing, Communication, Control and Automation (ICCUBEA)* (pp. 1-6). IEEE.

[16] Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., ... & Gui, T. (2023). The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

[17] Ko, K., Nyein, T. Y., Oo, K. K., Oo, T. Z., & Zin, T. T. (2024, November). Retrieval Augmented Generation for Document Query Automation using Open source LLMs. In *2024 5th International Conference on Advanced Information Technologies (ICAIT)* (pp. 1-6). IEEE.

[18] Mishra, P., Mahakali, A., & Venkataraman, P. S. (2024, August). SEARCHD-Advanced Retrieval with Text Generation using Large Language Models and Cross Encoding Re-ranking. In *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)* (pp. 975-980). IEEE.

[19] https://github.com/langchain-ai/rag-from-scratch

[20] Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., ... & He, Y. (2025). DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*