2022dc04341-jaykumar Chaudhary . .

2022dc04341

S2-24_DSECLZG628T - Mid

S2-24_DSECLZG628T

Birla Institute of Technology and Science Pilani WILP Division

Document Details

Submission ID **trn:oid:::1:3290129687**	**36 Pages**
Submission Date **Jul 4, 2025, 5:22 PM GMT+5:30**	**8,763 Words**
Download Date **Jul 4, 2025, 5:28 PM GMT+5:30**	**58,136 Characters**
File Name **2022dc04341.pdf**	
File Size **838.9 KB**	

6% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

42 Not Cited or Quoted 6%
Matches with neither in-text citation nor quotation marks

2 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

5%	Internet sources
1%	Publications
4%	Submitted works (Student Papers)

Match Groups

42 Not Cited or Quoted 6%
Matches with neither in-text citation nor quotation marks

2 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

5%	Internet sources
1%	Publications
4%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	**Birla Institute of Technology and Science Pilani**	**2%**
2	Internet	**www.mdpi.com**	**<1%**
3	Student papers	**ESoft Metro Campus, Sri Lanka**	**<1%**
4	Internet	**arxiv.org**	**<1%**
5	Internet	**orca.cardiff.ac.uk**	**<1%**
6	Internet	**repository.up.ac.za**	**<1%**
7	Student papers	**University of Hertfordshire**	**<1%**
8	Student papers	**University of New South Wales**	**<1%**
9	Publication	**Dagoberto Castellanos-Nieves, Luis García-Forte. "Human-Centered AI for Migran...**	**<1%**
10	Student papers	**Manchester Metropolitan University**	**<1%**

11	Publication	Pradeep Singh, Balasubramanian Raman. "Deep Learning Through the Prism of T...	<1%
12	Student papers	Flinders University	<1%
13	Publication	Trishit Mukherjee, Soumitra Sinhahajari, Debargha Mukherjee, Hrishikesh Mallic...	<1%
14	Student papers	ICTS	<1%
15	Student papers	Liverpool John Moores University	<1%
16	Publication	Wenli Yang, Lilian Some, Michael Bain, Byeong Kang. "A comprehensive survey o...	<1%
17	Internet	consulta-europa.com	<1%
18	Internet	repository.library.noaa.gov	<1%
19	Internet	123dok.net	<1%
20	Student papers	University of Queensland	<1%
21	Internet	1library.net	<1%
22	Internet	cupdf.com	<1%
23	Internet	docs.openvino.ai	<1%
24	Internet	researchportaltest.northumbria.ac.uk	<1%

25	Internet	
thesai.org		<1%
26	Publication	
Navin Sabharwal, Amit Agrawal. "Hands-on Question Answering Systems with BE...		<1%
27	Internet	
iris.unibas.it		<1%
28	Internet	
polynoe.lib.uniwa.gr		<1%
29	Internet	
www.preprints.org		<1%
30	Internet	
www.r4d.org		<1%

A Retrieval-Augmented Question Answering System Using BERT for Stream-Specific Educational Content

Submitted in partial fulfilment of the
requirements of the Degree: M.Tech in Data science and engineering

By

Jaykumar Chaudhary

2022DC04341

Under the supervision of

Mr. Lalkar Eknath Chhadawelkar

Technical Evangelist (Cybage Software, Pune)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

Pilani (Rajasthan) INDIA

JULY, 2025

Acknowledgement

I would like to express my heartfelt gratitude to all those who have supported and guided me throughout the journey of this dissertation.

First and foremost, I am deeply thankful to my supervisor, **Mr. Lalkar Eknath Chhadawelkar**, for his invaluable guidance, continuous encouragement, and patient supervision at every stage of this project. His insightful feedback and expertise have been crucial in shaping my work and motivating me to achieve my best.

I would also like to sincerely thank **Mrs. Radhika Raghuvanshi** for her initial guidance and for helping me refine the core idea of my dissertation. Her suggestions on the overall roadmap and her clarity of thought gave direction to my efforts from the very beginning.

My special thanks to **Jayalakshmi Natarajan ma'am**, my BITS evaluator, for her thoughtful evaluation and constructive comments, which have helped me strengthen both my research and its presentation.

Finally, I owe my deepest appreciation to my family for their unwavering support, constant encouragement, and understanding. Their faith in me has been my greatest source of motivation throughout this academic journey.

To all those who have helped me, directly or indirectly, in completing this work, I extend my sincere thanks.

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

SECOND SEMESTER 2024-25

DSECLZG628T / AIMLCZG628T DISSERTATION

CERTIFICATE

This is to certify that the Dissertation entitled A Retrieval-Augmented Question Answering System Using BERT for Stream-Specific Educational Content

and submitted by Mr. Jaykumar Chaudhary ID No. 2022DC04341 in partial fulfilment of the requirements of DSECLZG628T Dissertation, embodies the work done by him/her under my supervision.

BITS ID No. 2022DC04341 **Name of Student:** Jaykumar Chaudhary

Name of Supervisor: Mr. Lalkar Eknath Chhadawelkar

Designation of Supervisor: Technical Evangelist

Qualification and Experience: M.Tech in Data Science and Engineering, 28 Years

Official E- mail ID of Supervisor: lalkar@cybage.com

Topic of Dissertation: A Retrieval-Augmented Question Answering System Using BERT for Stream-Specific Educational Content

J. C. Chaudhary

Signature of Student

Date: 4-JULY-2025

Signature of Supervisor

Date: 4-JULY-2025

Abstract

In academic environments, students often struggle to quickly locate specific information across large, text-heavy educational documents such as course handouts. As elective subjects vary across semesters, navigating this material can be overwhelming, especially when trying to understand topics or explore subjects before making elective decisions. Traditional keyword-based search methods lack contextual understanding, leading to imprecise and inefficient information retrieval.

This dissertation proposes the design and development of a **Retrieval-Augmented Question Answering (QA) system** that leverages a **pretrained BERT-based model** to enable students to query academic content in natural language and receive accurate, contextually relevant answers. The system is primarily designed to process structured and semi-structured educational content — including course handouts — from the **Data Science** and **AI/ML streams**.

The proposed solution follows a **Retrieval-Augmented Generation (RAG) architecture** that separates the QA process into two components: **retrieval** and **reading**. First, a **retriever** uses **Sentence-BERT** to generate **embeddings** and identify relevant content, which is then indexed using **FAISS** for efficient **semantic search**. Next, a **reader model (BERT fine-tuned on SQuAD)** extracts the most likely answer span from the retrieved text.

Unlike conventional QA systems that require domain-specific fine-tuning or extensive labeled data, this system uses pretrained components and unsupervised document chunking. This makes it scalable, adaptable, and ideal for academic use. The system is designed to help students retrieve reliable answers to subject-related queries, assist in exam preparation, and support elective planning by giving clarity on subject depth and focus.

The expected result is a functional prototype capable of answering factual, definition-based, and conceptual questions using curriculum-aligned educational content. The system will be evaluated through QA metrics like **Exact Match** and **F1 Score**, along with qualitative feedback from users. This dissertation also discusses limitations such as the lack of support for **multi-hop reasoning** or generative answers. Nonetheless, it showcases a practical application of **NLP** in the academic domain, offering an intelligent way to support self-directed learning.

Key Words

BERT, Question Answering, Retrieval-Augmented Generation, Semantic Search, Educational NLP, Sentence-BERT, FAISS, Extractive QA

List of Symbols & Abbreviations

Symbol / Abbreviation	**Description**
NLP	Natural Language Processing
QA	Question Answering
RAG	Retrieval-Augmented Generation
LLM	Large Language Model
BERT	Bidirectional Encoder Representations from Transformers
SQuAD	Stanford Question Answering Dataset
FAISS	Facebook AI Similarity Search
ChromaDB	Chroma Vector Database
API	Application Programming Interface
PDF	Portable Document Format
DOCX	Microsoft Word Open XML Document Format
UI	User Interface
ID	Identification Number

List of Tables

List of Figures

18

21

6

Table of Contents

Chapter 1
Introduction

1.1 The Challenge of Academic Information Access

In the digital age, students face an overwhelming abundance of learning resources—PDF handouts, lecture notes, assignments, and supplementary materials—often spanning hundreds or thousands of pages. Paradoxically, this wealth of information can make it harder to locate precise, contextually accurate answers to academic questions. Traditional keyword-based search tools often miss nuanced relationships and fail to interpret mathematical or conceptual context, rarely surfacing the best answer.

Modern advancements in Natural Language Processing (NLP) offer new hope. **Retrieval-Augmented Generation (RAG)** stands out for its ability to blend semantic search with sophisticated answer generation, enabling direct, context-rich responses to complex questions. Rather than matching keywords, RAG systems retrieve the most relevant information from large document collections and use large language models (LLMs) to synthesize answers in natural language—a paradigm now transforming domains like enterprise search, medical research, and education.

This dissertation is dedicated to applying and advancing RAG methodologies for students and educators—demonstrating both the classic approaches and the new generation of modular, LLM-powered educational RAG systems.

1.2 Problem Statement

Despite having access to a wealth of digital academic materials, students often find it difficult to quickly and accurately extract relevant answers from large and complex documents. Existing keyword-based search tools lack the semantic understanding required for academic content, resulting in inefficient retrieval and superficial results. There is a clear need for an intelligent, context-aware question answering system that can understand student queries and deliver precise, explainable answers directly from their own course resources.

1.3 Motivation and Project Evolution

This project was envisioned as a way to transform the way students interact with their academic resources, empowering them to ask natural questions and receive precise, well-contextualized answers from their own course materials. The journey began with the implementation of a classic question-answering system—employing semantic chunking, Sentence-BERT embeddings, and BERT-based answer extraction—which successfully demonstrated that context-aware retrieval and QA are feasible within the educational domain.

Building on this solid groundwork, the project embraced the rapidly advancing NLP landscape. New frameworks such as **LangChain**, scalable vector databases like **ChromaDB**, and highly capable LLMs (such as **Google Gemini** and **Groq**) presented exciting opportunities to further enrich and extend the system. Integrating these tools enabled the development of a truly modular, next-generation **Educational RAG pipeline**—one that offers even greater flexibility, scalability, and answer quality.

Rather than a shift due to limitation, this transition represents a deliberate and strategic enhancement—leveraging the best of both classic and cutting-edge approaches to maximize the value delivered to students and educators.

1.4 System Overview

4 The following diagrams illustrate both the baseline and advanced architectures implemented in this project. This project implements a modern, modular Retrieval-Augmented Generation (RAG) application for educational question answering. The RAG methodology blends efficient document retrieval with advanced language generation, enabling the system to deliver precise, context-aware answers tailored to user queries.

The **high-level workflow** of the system is outlined in Figure 1 below:

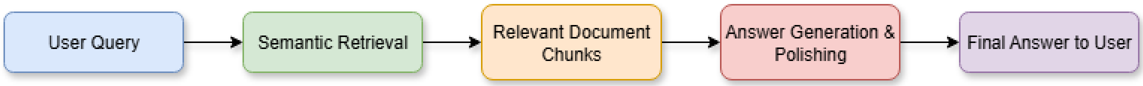


Figure 1: RAG Question Answering Pipeline

2 As shown, the process begins when a user submits a question through the application interface. The system performs semantic retrieval to identify the most relevant passages from
2 its academic knowledge base. These selected chunks are then passed to a large language model (LLM), which generates and polishes an answer using both the user's query and the supporting document context. The final, user-ready answer is then presented back to the user.

This streamlined RAG workflow underpins the flexible and robust academic QA capabilities discussed in greater depth throughout Chapters 3 and 4.

1.5 Objectives of the Dissertation

17 The overarching aim of this dissertation is to **bridge the gap between static academic documents and dynamic, question-driven learning**. Specifically, the project pursues the following objectives:

1. **To design and implement a scalable, semantic question-answering system** that enables students to ask open-ended questions and receive precise, well-contextualized answers sourced directly from their own course handouts, lecture notes, or study materials.

2. **To establish a technical and experimental comparison** between a classic, BERT-based QA pipeline and a modern, modular RAG system using cutting-edge tools (LangChain, ChromaDB, Gemini, Groq). The project will not only show how each system operates, but why the latter is more suitable for the challenges of educational QA at scale.

3. **To address key academic-specific NLP challenges**:

   - *Document chunking and context*: Developing chunking strategies that preserve the logical structure of academic texts, including mathematical notation and hierarchical headings.

   - *Semantic retrieval*: Ensuring the most relevant, contextually appropriate information is surfaced for each question, even across multiple handouts or complex documents.

   - *Answer trustworthiness and traceability*: Enabling answers to be sourced and referenced back to specific locations in academic documents.

4. **To create a reproducible, extensible codebase and application** that can be deployed by educators or students, integrating new models or document types with minimal changes.

5. **To evaluate the impact of advanced Educational RAG** in practical use, both through technical metrics (retrieval accuracy, answer quality, latency) and user-centric feedback.

Chapter 2
Literature Survey

2.1 Introduction

The landscape of question answering (QA) systems has evolved rapidly with advancements in natural language processing and deep learning. Traditional information retrieval systems—relying on keyword matching, bag-of-words models, or statistical ranking—could rarely capture the rich context and deep semantics inherent in complex academic materials. As educational resources have grown in scale and complexity, students increasingly need tools that can understand context, reason over information, and generate precise, trustworthy answers. Retrieval-Augmented Generation (RAG) addresses this need by combining the power of modern information retrieval with the generation capabilities of large language models (LLMs). This dual approach enables systems to first identify the most relevant parts of vast document collections and then generate high-quality, contextually aware answers that are both accurate and explainable—qualities essential for educational applications where trust and traceability matter (Liu et al., 2023; Bo Ni et al., 2025; Shah et al., 2024).

2.2 Classic RAG and BERT-Based QA

Classic RAG pipelines represent a significant leap forward from earlier QA architectures. At their core, these systems operate in two tightly integrated stages: retrieval and reading. In the retrieval stage, a user's question is converted into a dense vector representation, often using models such as Sentence-BERT. This vector is then used to search a vector database—commonly implemented with scalable tools like FAISS—to identify and rank the most semantically relevant document chunks (Liu et al., 2023; Wenqi Fan et al., 2024). The use of dense retrieval allows the system to capture subtle relationships, such as synonyms or rephrased concepts, that keyword search would miss.

Once the top-ranked chunks are retrieved, the reader stage utilizes a powerful pre-trained or fine-tuned BERT model to extract the specific answer span. This model can process entire passages, weighing both the immediate question and the broader context, to find the most likely answer. This two-step process—first narrowing the search space and then drilling down to the answer—delivers both efficiency and accuracy, making it highly effective for academic QA where precision is critical and answers are often embedded in dense, technical text (Wenqi Fan et al., 2024; Liu et al., 2023).

2.3 Advances in Chunking and Embedding

Chunking and embedding are foundational steps in any RAG system, but their importance is magnified in the educational domain. Academic materials are typically structured around complex, multi-part explanations, hierarchical headings, bulleted lists, and embedded formulas. Basic approaches that divide text by arbitrary length or page breaks risk fragmenting crucial context, leading to poor retrieval performance. Recent research and practical experiments advocate for **semantic chunking**: grouping together related explanations, definitions, and associated mathematical content so that each chunk represents a coherent knowledge unit (Franklin Lee & Tengfei Ma, 2025; Shah et al., 2024).

These carefully constructed chunks are then embedded into a high-dimensional vector space using advanced models like FastEmbed or domain-adapted Sentence-BERT. Embeddings capture both local and global context, enabling the system to match queries with the most appropriate content, even if it is phrased differently or spread across multiple sections. Effective chunking and embedding ensure that answers are not only relevant but also pedagogically sound—providing students with both factual content and the necessary context for deeper understanding (M. Shah et al., 2024; Bo Ni et al., 2025).

2.4 Modern Modular RAG with LLMs

The newest generation of RAG systems capitalizes on the capabilities of large language models and the flexibility of modular orchestration frameworks. Unlike earlier extractive QA pipelines, these architectures can perform complex reasoning and synthesis—combining information from multiple sources and presenting it in clear, structured, and natural language. Modular tools such as **LangChain** allow developers to seamlessly connect document loaders, semantic chunkers, dense retrievers, and a variety of LLMs (including Gemini, Groq, and locally hosted Ollama) within a single pipeline (Sonal Prabhune & Donald J. Berndt, 2024; Petko Georgiev et al., 2024; Rohan Anil et al., 2023).

Scalable vector databases like **ChromaDB** ensure that even large, multi-document educational datasets can be efficiently indexed and queried. LLMs are not limited to extracting text—they can summarize, paraphrase, and even explain answers, which is particularly valuable in education where explanations often matter as much as facts. The modularity of these systems supports easy experimentation, integration of new models, and adaptation to different academic subjects or user requirements. Studies show that such modular RAG systems outperform traditional pipelines in both answer quality and user engagement, while also making it easier to maintain and upgrade the QA system over time (Sonal Prabhune & Donald J. Berndt, 2024; Franklin Lee & Tengfei Ma, 2025).

2.5 Summary

Overall, the evolution from classic BERT-based QA to advanced, LLM-driven RAG systems enables much richer, more flexible question answering in academic settings. Our project follows this trajectory, combining the best practices identified in recent literature with innovations in modularity and retrieval (Liu et al., 2023; Shah et al., 2024).

Chapter 3
Baseline RAG System (Traditional Approach)

3.1 Introduction

This chapter presents the baseline question answering pipeline developed as the foundation of this project. This approach leverages established retrieval-augmented generation (RAG) techniques, focusing on traditional methods for preprocessing, chunking, semantic retrieval, and answer extraction. By systematically transforming raw academic materials into structured, searchable units and employing proven models for both retrieval and reading, the baseline system establishes the core workflow upon which later advancements are built. The results and observations from this pipeline not only demonstrate the feasibility of retrieval-based QA in educational contexts but also highlight key limitations that motivate the development of a more advanced, LLM-driven solution in subsequent chapters.

3.2 Document Chunking & Pre-processing

A robust question answering system must first convert raw academic materials—whether in PDF or DOCX format—into structured, meaningful segments for downstream analysis. This process, known as pre-processing and chunking, is crucial for ensuring the system can accurately locate and extract relevant answers.

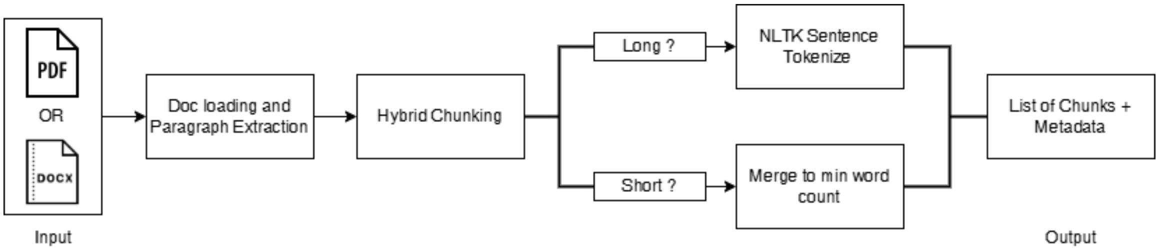


Figure 2: Preprocessing and Hybrid Chunking Workflow

This diagram illustrates the end-to-end workflow from input document (PDF or DOCX) through paragraph extraction and intelligent chunking, resulting in a collection of context-rich text segments ready for retrieval.

In this workflow, input documents are first loaded using appropriate tools (such as python-docx for DOCX files or a PDF parsing library for scanned materials). The system extracts non-empty paragraphs to ensure that only content-bearing text is processed. Each paragraph is then evaluated: unusually long paragraphs are split into sentences and regrouped using a hybrid strategy, while shorter paragraphs are merged to achieve optimal chunk size. This approach leverages both the document’s inherent structure and linguistic boundaries, resulting in chunks that balance context with focus.

The importance of this workflow lies in its direct impact on retrieval effectiveness. If chunks are too large, they may contain irrelevant or diluted information; if too small, they can lose essential context. By dynamically adjusting chunk size and boundaries, the system produces segments that are well-matched to the variety of queries posed by students.

One clear advantage of supporting both PDF and DOCX formats is the flexibility it offers in academic settings, where course materials can appear in a range of digital forms. The use of proven NLP libraries such as NLTK for sentence tokenization further enhances the quality of chunking, ensuring linguistic coherence within each segment.

A practical outcome of this preprocessing and chunking is shown below:

Chunk #	Word Count	Chunk Preview
1	96	Entropy is a measure of uncertainty in machine learning…
2	109	Regularization helps to prevent overfitting by introducing a penalty term…
3	80	The bias-variance tradeoff describes how model complexity affects prediction error…

Table 1: Chunks After Preprocessing and Hybrid Chunking

Through this approach, the system generates a curated list of searchable, informative text segments—laying a strong foundation for subsequent semantic embedding, retrieval, and answer extraction. This thoughtful chunking is essential for both the efficiency and accuracy of the overall QA pipeline, directly influencing the quality of answers provided to students.

3.2 Embedding & Retrieval

Accurate retrieval in question answering systems hinges on two fundamental processes: the conversion of text into meaningful vector representations, and the rapid search of these vectors to find relevant content. This section explains both aspects as implemented in the baseline pipeline.

3.2.1 Semantic Embedding with all-MiniLM-L6-v2

6

At the core of semantic search lies the **all-MiniLM-L6-v2 model**—a compact yet powerful transformer architecture. This model consists of six transformer encoder layers and roughly 22.7 million parameters. It is specifically optimized to generate fixed-length, 384-dimensional vector embeddings for sentences and short paragraphs, efficiently capturing their semantic content.

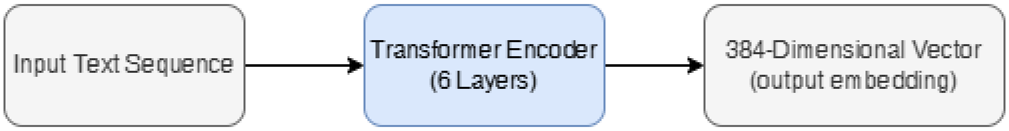

Input Text Sequence
Transformer Encoder (6 Layers)
384-Dimensional Vector (output embedding)

Figure 3: all-MiniLM-L6-v2 Model Architecture

The model is widely recognized for its ability to encode complex meaning into a small vector, making it ideal for large-scale semantic search. The resulting embeddings are robust to variations in wording and can be compared using simple distance metrics.

When a text chunk is processed by all-MiniLM-L6-v2, the output is a dense vector that can be directly compared with embeddings of other chunks or queries. The first few components of such vectors for sample chunks are shown in Table 2:

Chunk #	Chunk Preview	Embedding (First 8 Dims)
1	"Entropy is a measure…"	[0.022, -0.108, 0.176, …]
2	"Regularization helps…"	[0.073, 0.029, 0.109, …]

Table 2: Embedding Vectors for Text Chunks

3.2.2 Efficient Retrieval with FAISS

A central challenge in large-scale question answering is finding the most relevant text chunks for any given query—quickly and accurately—among thousands of candidates. This is where **FAISS (Facebook AI Similarity Search)** becomes essential. Developed by Meta AI, FAISS is an open-source library purpose-built for fast similarity search and clustering of dense vectors, making it especially suitable for natural language processing and semantic search applications.

FAISS excels at handling high-dimensional embeddings, such as those produced by neural language models, and can scale to collections containing millions of vectors. It offers a range of indexing strategies, from brute-force (exact) search to highly efficient approximate methods, enabling the system designer to balance accuracy and latency as needed. For academic QA, FAISS is typically used with normalized embeddings and inner product (cosine similarity) search, ensuring that semantically similar text chunks are identified regardless of their original scale or distribution.

Several factors make FAISS the preferred choice for this system:

- **Speed and Scalability:** FAISS supports real-time search, delivering sub-second responses even as the academic corpus expands.
- **Flexibility:** It provides multiple search algorithms suitable for both small and very large datasets.
- **Ease of Integration:** FAISS is compatible with major machine learning environments and is easy to deploy in Python-based pipelines.
- **Reliability:** As a widely adopted open-source project, FAISS is robust and production-ready.

In this pipeline, all chunk embeddings are added to a FAISS index configured for inner product search. When a user question arrives, it is embedded using the same model and submitted to FAISS, which instantly returns the top-k most semantically similar chunks from the entire collection.

The effectiveness of FAISS retrieval is illustrated by the sample queries and their best-matched chunks in Table 3:

Query	Top Chunk (Preview)	Similarity Score
What is entropy in ML?	Entropy is a measure of uncertainty...	0.87
How do we prevent overfitting?	Regularization helps prevent...	0.83

Table 3: FAISS Retrieval Results for Sample Queries

By leveraging FAISS, the retrieval module ensures that user queries are rapidly and accurately matched with the most relevant academic content, providing a robust foundation for high-quality answer extraction downstream.

3.2.3 Implementation Workflow: Embedding and Retrieval

The embedding and retrieval components of the baseline question answering system are designed to operate in a streamlined, reproducible workflow. This architecture ensures that academic materials are consistently processed and indexed, while queries are efficiently matched to relevant content—regardless of scale or document format.

At the implementation level, the workflow begins with the collection of pre-processed text chunks obtained from various academic documents. Each chunk is systematically fed into the all-MiniLM-L6-v2 embedding model, resulting in a dense vector representation that encapsulates its semantic meaning. These vectors are normalized and added to a FAISS index, forming the searchable database.

20 When a user submits a question, it undergoes the same embedding process, producing a query vector in the same high-dimensional space. This query embedding is then used to search the FAISS index, which rapidly returns the top-k most semantically similar document chunks. These retrieved chunks are passed to the downstream answer extraction stage.

This modular pipeline not only ensures high performance and scalability, but also supports easy extension and debugging—each stage is logically separated and can be independently evaluated.

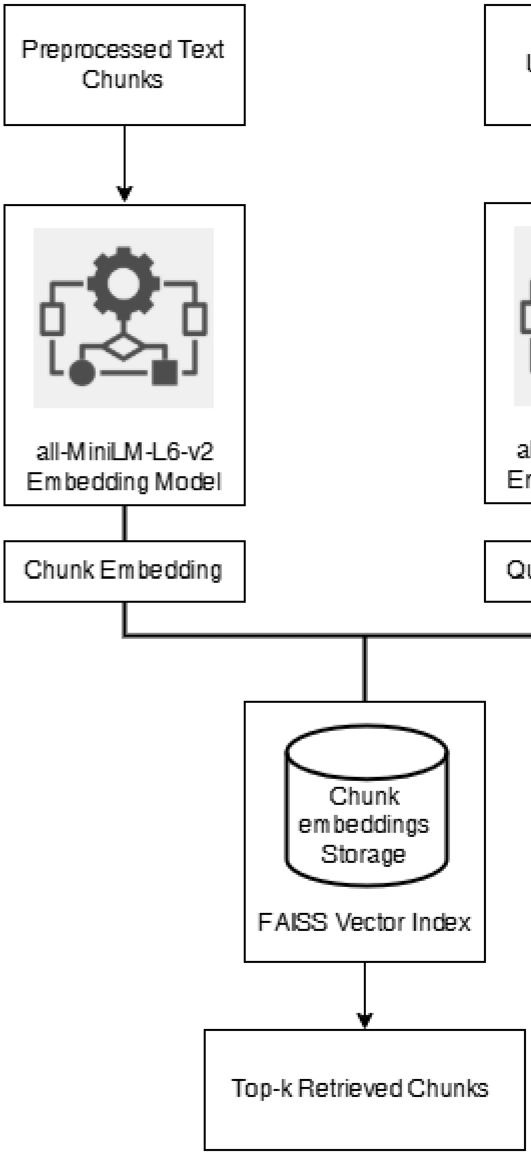


Figure 4: Implementation Flow of Embedding and Retrieval Pipeline

This diagram depicts the system's operational steps, from input documents through embedding, indexing, and retrieval, with clear indication of data flow between each stage.

3.3 Reader Module

The final step in the baseline pipeline is answer extraction—identifying and presenting the most relevant, concise response to a user's question. This is accomplished through a two-stage approach: first, a state-of-the-art BERT-based reader is applied to the top retrieved text chunks, and second, the system employs smart keyword-based heuristics and fall-back strategies to maximize answer reliability, even when the direct extractive model is uncertain.

3.3.1 Answer Extraction with BERT QA Model

The reader module in this system is built upon the **bert-large-uncased-whole-word-masking-finetuned-squad** model, one of the most robust and widely adopted architectures for extractive question answering. BERT (Bidirectional Encoder Representations from Transformers) is a deep transformer-based language model designed to understand context in natural language by processing words in both directions—left-to-right and right-to-left—within a sentence.

The **"bert-large-uncased-whole-word-masking-finetuned-squad"** variant is a large-scale model featuring 24 transformer encoder layers and over 340 million parameters. It is pre-trained on a vast English corpus and further fine-tuned on the SQuAD (Stanford Question Answering Dataset), specializing it for reading comprehension and QA tasks. The “whole word masking” technique during training enables BERT to develop a more nuanced sense of word boundaries and contextual relationships, improving answer extraction accuracy.

In this pipeline, when a user submits a question, it is paired with the top retrieved chunk from the document collection. The BERT QA model receives both as input and processes them through its deep stack of transformer layers, leveraging multi-head self-attention mechanisms to model complex contextual dependencies. The model then outputs the most probable start and end positions for the answer span within the context, along with a confidence score for its prediction.

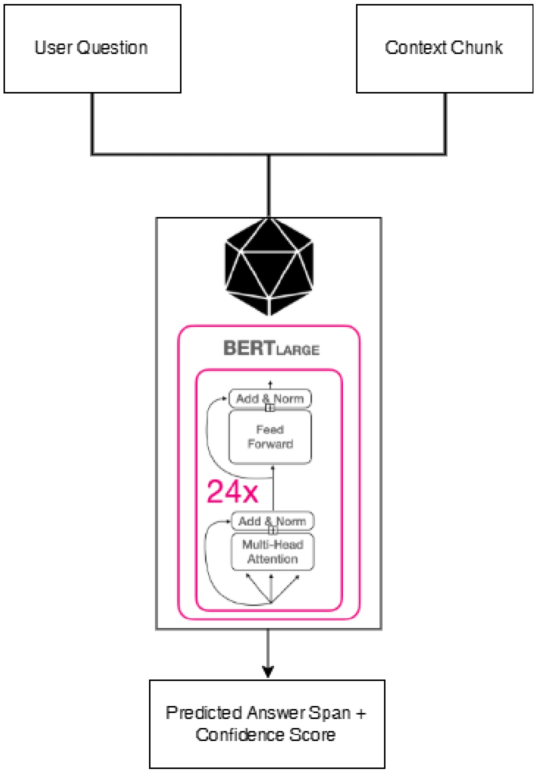


Figure 5: BERT-Large SQuAD QA Pipeline Architecture

The pipeline's workflow can be summarized as:

- **Input:** User question and top-ranked context chunk.
- **Model:** "bert-large-uncased-whole-word-masking-finetuned-squad" processes through multiple self-attention layers.
- **Output:** Predicted answer span within the context, with an associated confidence score.

This approach ensures that the system can accurately extract answers directly from text, particularly when the answer is explicitly stated in the context and closely matches the phrasing of the question. The result is a high-precision, explainable answer that can be traced to a specific document segment.

3.3.2 Handling Edge Cases: Keyword and Fall-back Heuristics

While transformer-based models like BERT excel at extracting answer spans when the context is explicit and well-matched, real-world academic handouts can present cases where answers are implicit, phrased differently, or distributed across multiple sentences. To ensure robustness, the pipeline incorporates a two-stage backup mechanism: **global keyword matching** and a final **fall-back heuristic**.

In the first backup stage, the system employs a keyword overlap strategy. When the BERT QA model's output does not meet minimum standards for answer length or confidence, the system scans the Top-K retrieved chunks for sentences sharing key non-stop word terms with the user's question. By tallying the intersection of content words between the question and each candidate sentence, the method surfaces sentences that, while perhaps not extracted by the neural model, are likely to contain directly relevant information.

If this keyword matching still fails to yield a suitable answer, the pipeline defaults to a straightforward fall-back: presenting the first sentence of the top-ranked chunk. This ensures the user always receives a response grounded in the most relevant context available, even if the answer is only partially matched.

This layered approach—moving from neural extraction, to keyword scoring, to context-based fall-back—ensures answer delivery is both robust and explainable. It also supports transparency for users and evaluators, as each answer can be traced to the logic or source that produced it.

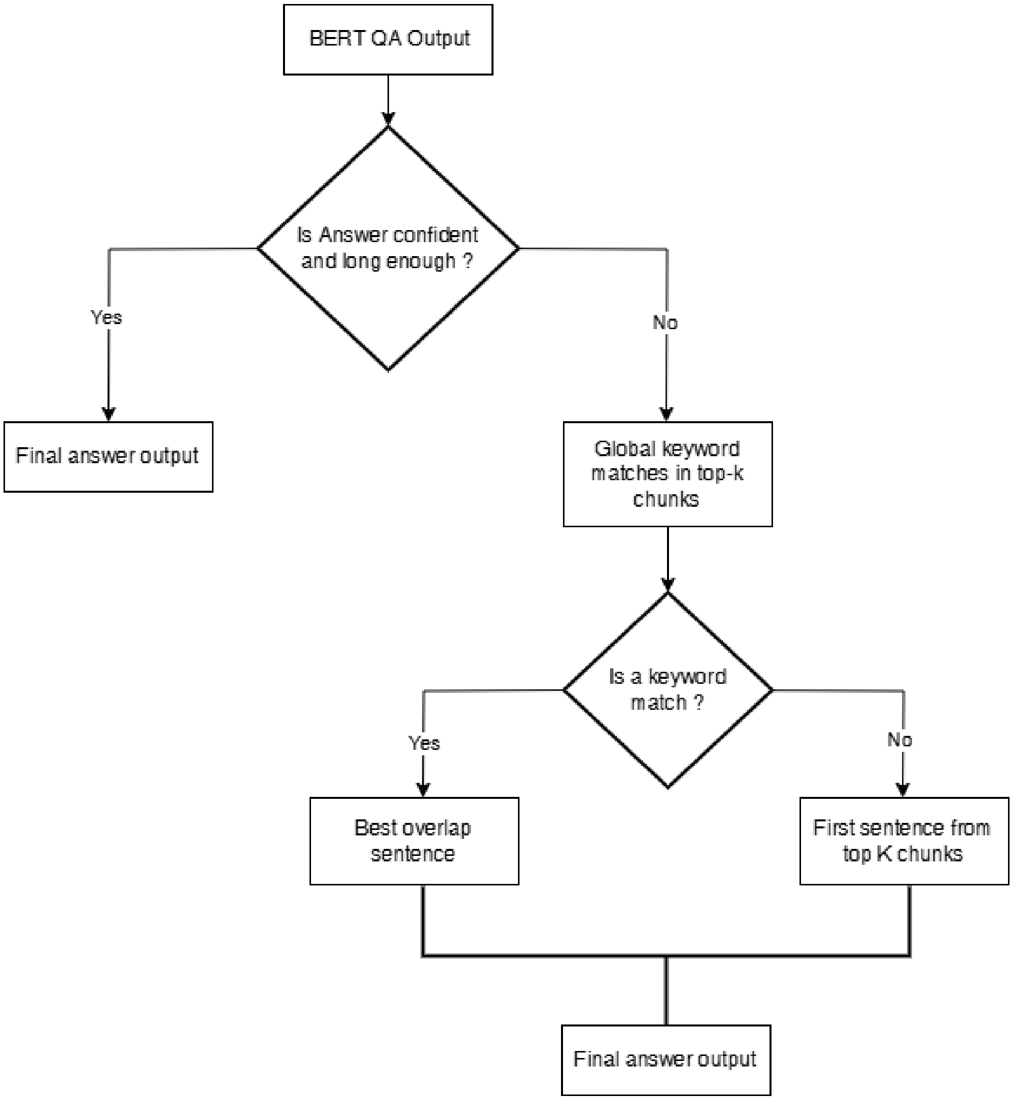

BERT QA Output
Is Answer confident and long enough ?
Yes
No
Final answer output
Global keyword matches in top-k chunks
Is a keyword match ?
Yes
No
Best overlap sentence
First sentence from top K chunks
Final answer output

Figure 6: Decision Flow for Reader Module Edge Cases

To illustrate the decision process and variety of answer sources, Table 4 shows example queries with the answer produced, the method used, and any confidence or keyword overlap scores as available.

Query	Answer	Method Used	Score/Type
What is overfitting?	"Overfitting occurs when a model…"	BERT QA	0.93
What are optimization methods?	"Gradient descent is a commonly…"	Keyword Match	Overlap: 3 words
Explain variance in ML.	"Variance is the amount by which…"	Fallback	n/a

Table 4: Example of Answer Sources and Types in Edge Cases

This multi-tiered reader module design ensures high answer reliability, maximizing the chance of providing a relevant, complete response for a broad spectrum of academic questions.

3.4 Observations & Limitations

Evaluation of the baseline retrieval-augmented question answering pipeline highlights several practical strengths and key limitations in handling complex academic documents.

Key Observations:

- The combination of **hybrid chunking** (merging and splitting paragraphs/sentences by length), dense embeddings (all-MiniLM-L6-v2), and FAISS-based retrieval enables the system to surface relevant passages, even when query phrasing differs from the source.
- The BERT-large SQuAD reader shows strong accuracy for fact-based questions where answers are clearly stated within the top retrieved chunk.
- Keyword-based and fall-back heuristics provide robustness, ensuring the system delivers some answer even when the neural model's confidence is low.

Limitations and Challenges:

- **Context Fragmentation:** Only one chunk is used for answer extraction, so questions requiring information across multiple sections or paragraphs are not well-supported.
- **Extractive-only Answers:** The pipeline can only extract literal spans; it cannot synthesize or rephrase for explanatory or multi-part queries.
- **Chunking Strategy Sensitivity:** The effectiveness of retrieval and answer extraction depends heavily on how chunks are formed; improper chunk size or grouping may lead to lost context or irrelevant content.
- **Implicit and Scattered Answers:** When answers are implied or distributed across sentences, the system may fail to retrieve a satisfactory result.
- **Resource Scalability:** As the academic corpus grows, embedding and search operations become more demanding, even with FAISS optimizations.
- **Model Limitations:** Answer quality can be limited by the generalization and vocabulary coverage of the pre-trained models, particularly in specialized subject areas.

Chapter 4
Modern RAG System with LLMs & Ecosystem Tools

Recent advancements in large language models and the open-source NLP ecosystem have opened new horizons for academic question answering. This chapter introduces an advanced, modular Retrieval-Augmented Generation (RAG) pipeline that leverages cutting-edge technologies—including LangChain for orchestration, ChromaDB for vector storage, and LLMs such as Gemini and Groq—for a new level of performance, flexibility, and user experience. The chapter contrasts this approach with the traditional baseline and demonstrates how these innovations address key challenges observed in the classic pipeline.

4.1 Motivation for Advancing the Pipeline

While the traditional retrieval-augmented QA pipeline provides a strong foundation for academic search, its limitations become increasingly apparent as the complexity and scale of educational content grow. Three primary challenges drive the motivation for a modernized, LLM-centric architecture:

1. Scalability and Flexibility:

The classic pipeline relies on static chunking and embedding workflows, which become cumbersome as the volume and diversity of course materials increase. Adapting to new document formats, custom chunking logic, or the addition of new data sources often requires manual intervention or pipeline reconfiguration.

2. Limitations of Extractive QA:

BERT-based models, though powerful for extractive tasks, are inherently limited to returning verbatim spans from retrieved chunks. They struggle with open-ended, generative, or multi-hop questions, and cannot synthesize information from across multiple passages.

3. Advances in LLMs and Tooling:

The emergence of highly capable large language models—such as Gemini, Groq, and locally-deployable models via Ollama—enables true generative QA, where answers can be synthesized, summarized, or reworded as needed. At the same time, ecosystem tools like LangChain and ChromaDB offer modular, extensible building blocks for robust retrieval, prompt chaining, and integration with multiple LLMs, making the system easier to scale and maintain.

These factors collectively motivate the transition to a **modular, LLM-driven RAG architecture**—one designed to overcome the limitations of the traditional baseline, adapt to diverse academic domains, and deliver context-aware, high-quality answers at scale.

4.2 New Architecture Overview

The modern RAG system developed in this work is designed to overcome the bottlenecks and rigidity of the traditional QA pipeline. Its architecture is highly modular, scalable, and ready for future enhancements—leveraging leading open-source ecosystem tools and best-in-class large language models.

Key components include:

- **LangChain:** Orchestrates the pipeline, managing everything from document loading and chunking to multi-stage prompt chaining and retrieval.
- **ChromaDB:** Provides persistent, high-performance vector storage, enabling rapid and scalable semantic search across large and dynamic document collections.
- **LLM Reader (Gemini, Groq):** Acts as the answer generator, producing not only extractive but also generative and explanatory responses. These LLMs can synthesize information, summarize across documents, and deliver richer answers.
- **Ollama (Future Scope):** The architecture is explicitly designed to accommodate integration with locally hosted LLMs (such as those available via Ollama), supporting privacy-preserving and on-premises deployments for academic institutions.
- **Pluggable Embedding Back-ends:** The pipeline supports easy switching of embedding models or APIs, ensuring adaptability as the NLP landscape evolves.

4.2.1 System Design and Data Flow

This modular architecture enables each pipeline component to be independently developed, tuned, or upgraded. Academic documents in various formats are ingested, chunked, and embedded using advanced models, with vectors stored in ChromaDB for fast semantic retrieval. LangChain orchestrates retrieval workflows, supporting context-aware, multi-hop queries. Retrieved passages and user questions are processed by state-of-the-art LLM readers (Gemini, Groq), which generate explanatory or synthesized answers. The design also anticipates future integration of local LLMs via Ollama, ensuring privacy and control for institution-specific deployments. This foundation positions the system for rapid adaptation as new models and retrieval tools emerge.

Please refer to Figure 7 below for a high-level overview of the modern modular RAG pipeline architecture, illustrating the flow of data and modular integration of each component.

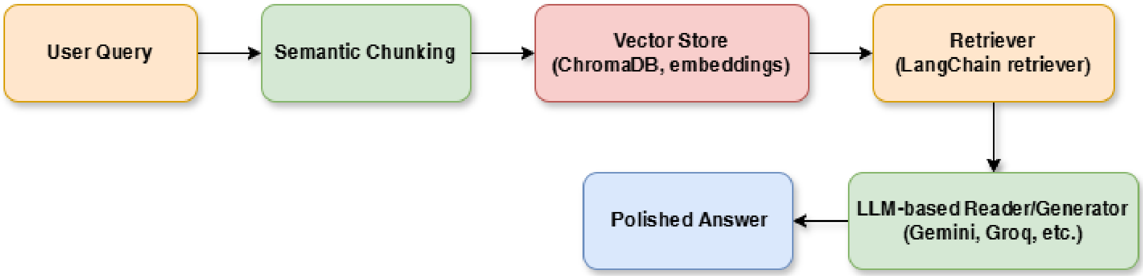


Figure 7: Overview of the modern modular RAG pipeline architecture

Table 5 below summarizes the key differences between the baseline and modern RAG architectures. The modern system offers improved scalability, flexibility, and answer quality through modular orchestration, persistent vector storage, and generative LLM capabilities, while also being future-ready for local LLM integration.

Feature	Baseline System	Modern RAG System
Orchestration	Hardcoded pipeline	LangChain modular flows
Vector Store	FAISS (in-memory)	ChromaDB (persistent)
Reader Model	BERT-large (extractive)	Gemini/Groq (LLM, generative), Ollama (future/local)
Document Formats	DOCX, PDF (manual)	Multi-format, extensible
Chunking Strategy	Hybrid size-based	Pluggable/context-aware
Retrieval Flexibility	Top-k only	Multi-hop, hybrid
Scalability	Limited	High
Answer Style	Extractive	Generative/explanatory
Integration Ease	Low (manual code)	High (configurable)

Table 5: comparison of the baseline and modern RAG pipeline architectures

4.3 Enhanced Retriever Module

The retriever module in the modern RAG pipeline represents a significant leap beyond traditional approaches, combining advanced semantic chunking, state-of-the-art embedding models, and persistent vector storage to deliver accurate, scalable, and context-aware retrieval. By integrating LangChain's flexible retriever framework, ChromaDB, and the latest in semantic embedding technology, the system supports robust document ingestion, flexible chunking strategies, and highly efficient search—laying a strong foundation for LLM-based answer generation in academic settings.

4.3.1 Semantic Chunking of Academic Documents

Semantic chunking is a transformative step that goes beyond basic length-based splitting or hybrid chunking. Rather than merely dividing text by word count or paragraph boundaries, semantic chunking aims to segment documents into coherent, meaning-rich passages that align with natural topic shifts and content boundaries. This ensures that each chunk is not only manageable in size but also maximally relevant for retrieval and answer generation.

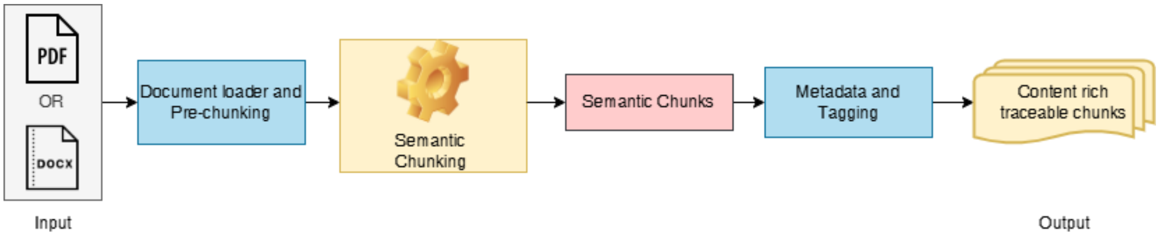


Figure 8: Semantic Chunking Workflow

In practice, the pipeline leverages **LangChain's SemanticChunker**, which utilizes a pre-trained embedding model to identify content boundaries based on semantic coherence rather than superficial markers. Each document is initially pre-chunked—often at the page or paragraph level—to simplify processing. The SemanticChunker then analyses these segments, using statistical or percentile-based thresholds on embedding similarity to determine where one idea ends and another begins.

This approach produces chunks that more closely match the way humans naturally navigate and reference academic material, leading to higher retrieval precision. Every chunk is tagged with rich metadata—including the source document, page number, and, if relevant, surrounding context—making it possible to trace every answer back to its origin.

A sample output from the semantic chunking stage is illustrated below:

Chunk #	Source PDF	Chunk Text Preview	Metadata (Page)
1	lecture1.pdf	Entropy is a measure of...	4
2	lecture2.pdf	Regularization prevents over...	7

Table 6: Example of Semantic Chunk Metadata

4.3.2 Advanced Embedding with FastEmbed (BAAI/bge-base-en-v1.5)

A pivotal step in the modern retriever pipeline is converting each semantic chunk into a vector that accurately represents its meaning for semantic search. This process, known as **embedding**, is essential for allowing the system to retrieve content based on conceptual similarity, rather than just matching keywords.

9 In this work, the **FastEmbed** framework, utilizing the **BAAI/bge-base-en-v1.5** model, is employed for this purpose. This state-of-the-art model produces 768-dimensional embeddings, designed to capture the core semantics of sentences and paragraphs—making it highly effective for academic QA over diverse content.

Before delving deeper, consider the flow illustrated in Figure 9 below:

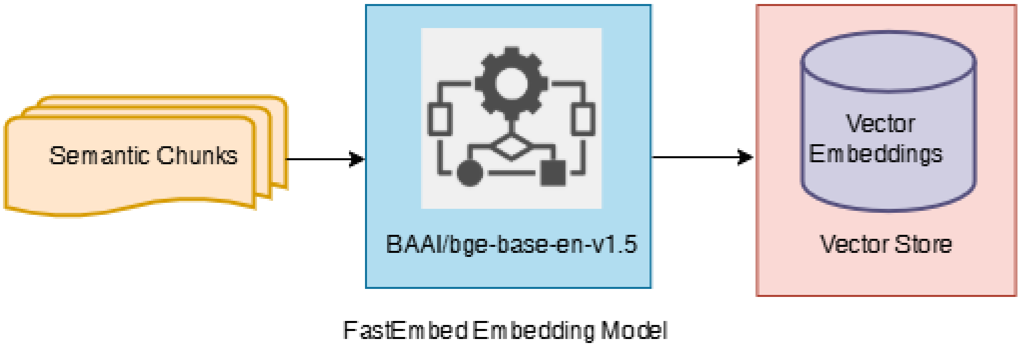


Figure 9: Embedding Workflow for Semantic Chunks

The process begins with the semantic chunks produced in the previous stage. Each chunk enters the embedding model, depicted in the center box. The FastEmbed (BAAI/bge-base-en-v1.5) model applies deep language understanding to encode the full meaning of the text—not just the words used—into a 768-dimensional vector. This is shown in the diagram by the downward arrow from "Semantic Chunks" into the "Embedding Model" box. The resulting output, a dense vector, is directed into the "Vector Embeddings" box, representing a collection of these embeddings for all chunks in the academic corpus.

This visual flow captures the essence of the embedding phase: it bridges raw, meaning-rich text with a mathematical space optimized for rapid, context-sensitive retrieval. By mapping all document chunks in this way, the system can efficiently compare user queries to the most semantically relevant passages, regardless of surface phrasing or document structure.

A sample of the generated embeddings is shown below:

Chunk #	Chunk Preview	Embedding (First 8 Dims)
1	Entropy is a measure of...	0.071, -0.021, 0.122, 0.043, -0.055, 0.108, 0.023, -0.037
2	Regularization prevents over...	0.091, -0.014, 0.113, 0.037, -0.051, 0.104, 0.021, -0.029

Table 7: Example Embeddings for Semantic Chunks

4.3.3 Persistent Vector Storage with ChromaDB

Once each semantic chunk is transformed into a dense vector embedding, an efficient and scalable method is needed to store, index, and retrieve these vectors for fast semantic search. This is achieved through the use of ChromaDB, a modern, open-source vector database optimized for high-throughput retrieval and persistent storage.

ChromaDB is designed to handle large-scale academic corpora, supporting rapid insertion, querying, and deletion of high-dimensional vectors. Its persistence feature allows the vector index to be saved and reloaded from disk, ensuring that the system can handle evolving document sets without the need to recompute all embeddings at each startup. This makes the pipeline suitable for real-world, production-scale deployments in educational settings.

Figure 10 below depicts the flow of data and vectors into ChromaDB:

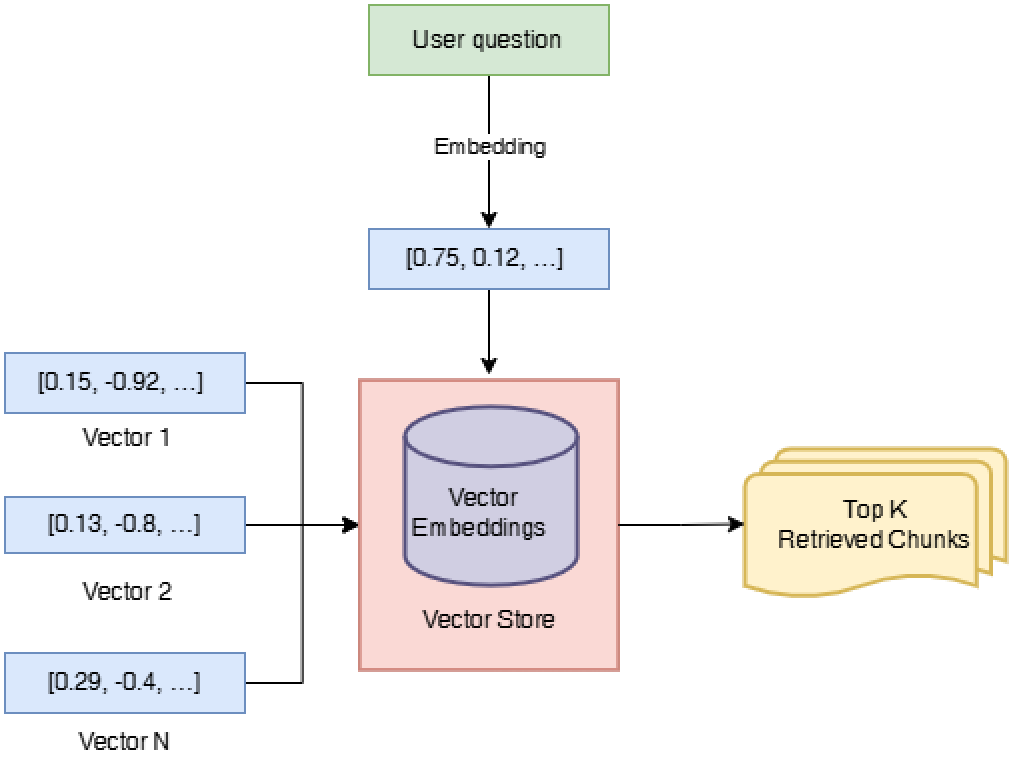


Figure 10: Document and query embeddings into ChromaDB

14

In this workflow, all chunk embeddings are stored in ChromaDB (center box). When a user submits a question, it is passed through the same embedding model and the resulting query vector is sent to ChromaDB. The vector store rapidly computes similarity scores and returns the top-k most similar document chunks for downstream LLM-based answer generation. This design not only speeds up retrieval, but also ensures persistence—vectors and metadata can be stored across sessions, and the system can scale to tens of thousands of documents.

Table 8 below provides an example of the indexed data for a few chunks:

Chunk ID	Source Document	Metadata (Page)	Vector Store Path
001	lecture1.pdf	4	/outputs/chroma_semantic...
002	lecture2.pdf	7	/outputs/chroma_semantic...

Table 8: Example Entries in ChromaDB Vector Store

4.3.4 Retriever Object Creation from the Vector Store

The culmination of the enhanced retriever module is the creation of a standardized retriever object, which acts as the interface for semantic search over the indexed document chunks. Once all semantic chunks have been embedded and stored in ChromaDB, this object is initialized to efficiently surface the most relevant passages in response to search queries or for downstream answer generation.

The underlying logic connects the persistent vector store containing the indexed semantic chunks to the retriever component responsible for executing top-k similarity searches. This modular design ensures flexibility and maintainability, making it easy to adjust retrieval parameters or integrate additional functionality as the pipeline evolves.

This process is illustrated below:

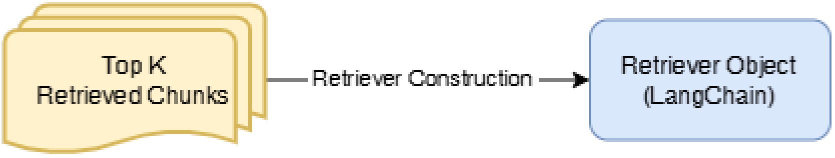


Figure 11: Construction of a retriever object from the ChromaDB vector index

By encapsulating the retrieval logic in this way, the modern RAG system enables robust, context-aware selection of document passages assuring that the most relevant academic content is efficiently delivered to the LLM-based answer generation module.

4.4 Advanced Reader/LLM Module

The advanced Reader module is a defining strength of the modern RAG pipeline. In this stage, generative large language models (LLMs) are leveraged to transform relevant retrieved content into fluent, user-friendly answers. The system currently supports integration with **Google Gemini (gemini-1.5-flash-latest)** and **Llama 3 (llama3-8b-8192) via Groq's API**, each bringing unique advantages to academic question answering.

Supported Language Models

- **Google Gemini (gemini-1.5-flash-latest):**

  Gemini is Google's state-of-the-art transformer-based LLM, known for fast inference and robust context handling. The "1.5 Flash" variant is optimized for cost-effective, rapid question answering, efficiently processing large prompts and generating high-quality answers. Its architecture is designed to extract and synthesize information from the provided context, making it highly effective for academic queries.

- **Llama 3 (llama3-8b-8192 via Groq):**

  Llama 3, an open-weight model from Meta, is accessed via Groq's ultra-fast, API-hosted inference platform. With 8 billion parameters and an 8,192-token context window, Llama 3 excels at generating coherent, detailed, and contextually appropriate answers, even when questions are nuanced or multi-faceted. The Groq backend offers extremely low-latency, making the QA experience more interactive and responsive.

4.4.1 LLM-Based Answer Generation Workflow

After the retriever provides the top-k most relevant document chunks, these passages, along with the user's question, are formatted into a prompt and submitted to the selected LLM. The initial LLM-generated answer is then passed through a dedicated polishing stage, which utilizes the **spaCy** natural language processing library for refinement. This polishing process may involve:

- Trimming redundant phrases,
- Removing irrelevant content,
- Standardizing terminology,
- Formatting the answer for clarity or length.

By leveraging spaCy, the system can perform advanced linguistic analysis, such as part-of-speech tagging and named entity recognition, to systematically clean and enhance the LLM output before presenting it to the user.

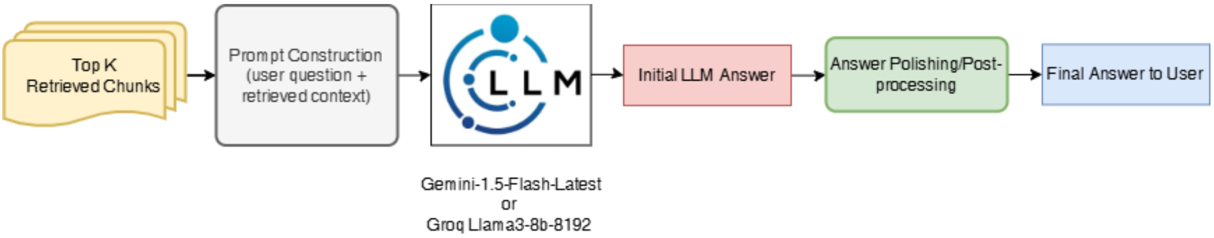


Figure 12: LLM-Based Answer Generation and Polishing Workflow

Query	LLM Backend	Final Polished Answer (Excerpt)
What is entropy in ML?	Gemini 1.5 Flash	"Entropy in machine learning measures the uncertainty or randomness present in data."
Explain regularization	Llama 3 (Groq, 8B-8192)	"Regularization is a technique used to prevent overfitting by adding a penalty to the loss function."

Table 9: Sample LLM answers after polishing.

4.5 Implementation Pipeline: End-to-End Workflow and Code Structure

A core strength of the modern RAG pipeline is its modular, maintainable implementation—where each major step in both indexing and question-answering is encapsulated in a dedicated Python module. The system leverages a web interface built with Streamlit, providing an intuitive user experience for both uploading academic resources and submitting queries. The diagram below (Figure 13) illustrates the complete process, showing how data and logic flow through the codebase for both document ingestion and QA.

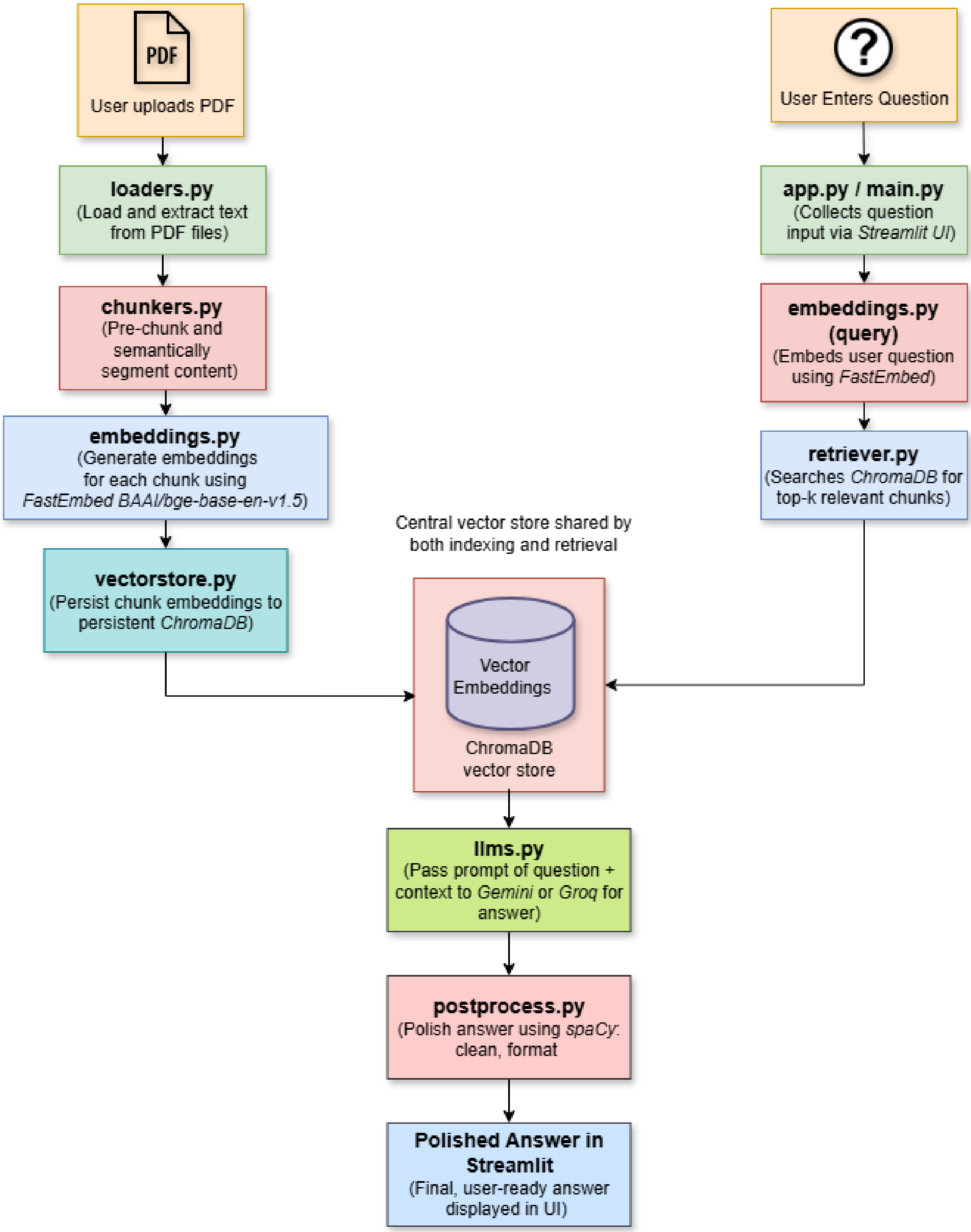

PDF
User uploads PDF
loaders.py
(Load and extract text from PDF files)
chunkers.py
(Pre-chunk and semantically segment content)
embeddings.py
(Generate embeddings for each chunk using FastEmbed BAAI/bge-base-en-v1.5)
vectorstore.py
(Persist chunk embeddings to persistent ChromaDB)
User Enters Question
app.py / main.py
(Collects question input via Streamlit UI)
embeddings.py (query)
(Embeds user question using FastEmbed)
retriever.py
(Searches ChromaDB for top-k relevant chunks)
Central vector store shared by both indexing and retrieval
Vector Embeddings
ChromaDB vector store
llms.py
(Pass prompt of question + context to Gemini or Groq for answer)
postprocess.py
(Polish answer using spaCy: clean, format
Polished Answer in Streamlit
(Final, user-ready answer displayed in UI)

Figure 13: Modular RAG pipeline implementation

Indexing Flow (Document Upload)

When a user uploads a new PDF (such as a course handout), the file is first processed by loaders.py, which extracts the raw text from the document. Next, chunkers.py pre-chunks and then semantically segments the content into context-rich passages. These chunks are embedded into high-dimensional vectors using FastEmbed via embeddings.py (with the BAAI/bge-base-en-v1.5 model). Finally, vectorstore.py stores all chunk embeddings in a persistent ChromaDB vector index. This index enables efficient, semantic retrieval of academic content, and persists across sessions for rapid reuse.

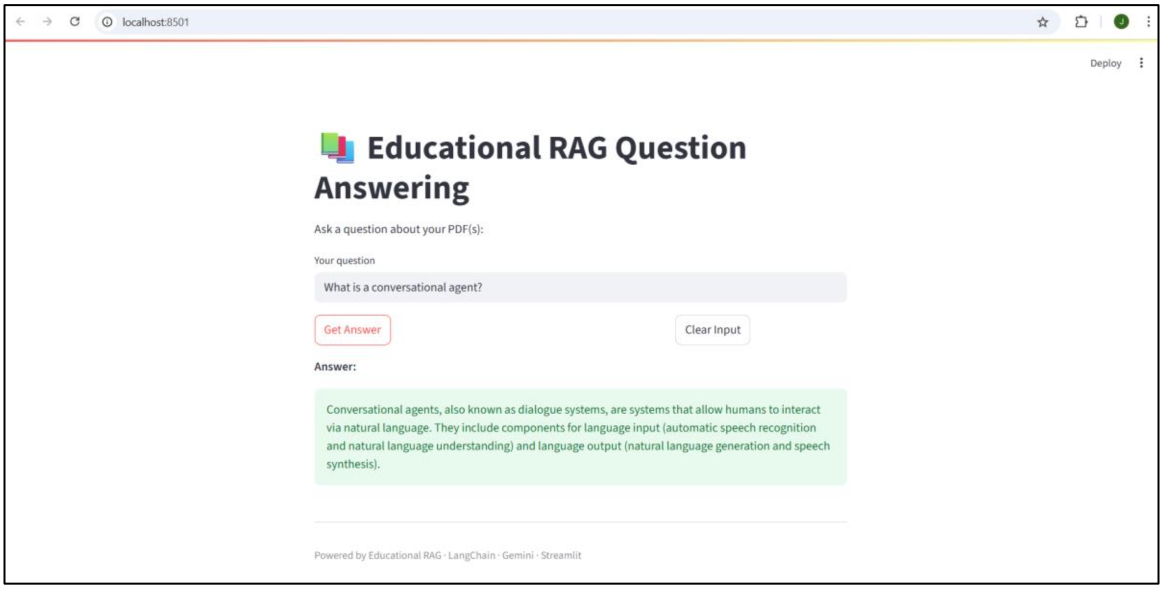
QA Flow (User Question)

When a student submits a question through the **Streamlit UI** (app.py/main.py), the system first embeds the question (again using embeddings.py for consistency). The retriever.py module then performs a semantic search in the ChromaDB index to identify the top-k most relevant document chunks. These, along with the user’s question, are passed to llms.py, which orchestrates answer generation using either the **Gemini or Llama 3 LLM** backend. The generated answer is then polished with **spaCy** in postprocess.py—cleaning, formatting, and ensuring clarity—before being displayed back to the user in the web interface.

This end-to-end architecture ensures that each function is isolated for easy maintenance and future upgrades. The clear, top-down flow means both developers and reviewers can readily trace how information moves from document ingestion all the way to a polished answer, ensuring transparency and reliability throughout the RAG pipeline.

User Interface Snapshot

The final user interface of the Educational RAG Question Answering system provides a clean, intuitive web experience for academic queries over uploaded PDFs. Built with Streamlit, the UI allows users to submit questions and instantly receive context-aware answers generated by the RAG pipeline—showcasing both accessibility and real-time performance.



Educational RAG Question Answering

Ask a question about your PDF(s):

Your question

What is a conversational agent?

Answer:

Conversational agents, also known as dialogue systems, are systems that allow humans to interact via natural language. They include components for language input (automatic speech recognition and natural language understanding) and language output (natural language generation and speech synthesis).

Powered by Educational RAG · LangChain · Gemini · Streamlit

Figure 14: User Interface Snapshot

Chapter 5
Progress and Future Directions

5.1 Work Completed till Mid-Semester

By the mid-semester milestone, the following progress had been achieved:

- **Developed and validated a traditional QA baseline** using Sentence-BERT embeddings, FAISS for retrieval, and a classic BERT reader for answer extraction.
- **Designed and implemented the advanced Educational RAG pipeline**, integrating modular chunking, embedding via FastEmbed, vector storage with ChromaDB, and answer generation using large language models such as Google Gemini and Groq—fully orchestrated with LangChain.
- **Built a user-facing demo application** enabling real-time academic QA over PDF documents.
- **Conducted initial evaluations** demonstrating improved flexibility, accuracy, and user experience with the LLM-based approach compared to the baseline.

5.2 Discussion and Path Forward

The modular RAG pipeline developed in this work serves as a strong foundation for academic question answering, but several practical and research-oriented directions remain for further enhancement. The following initiatives outline the immediate path forward:

1. Exploration and Evaluation of Ollama LLM Integration

The next phase will be to explore the integration of **Ollama** for running open-weight large language models locally, enabling on-premises, privacy-preserving answer generation alongside current Gemini and Groq integrations. This will include local setup, hands-on experimentation, and comprehensive benchmarking—comparing performance, accuracy, and efficiency with cloud-based LLMs. The system's modular architecture ensures robust fallback to cloud LLMs if technical or operational constraints arise, allowing us to rigorously determine the best-fit solution for various deployment scenarios.

2. Automated Question Generation from Academic Documents

To enrich both user engagement and evaluation, future versions will implement modules to automatically generate practice or assessment questions directly from the ingested academic content. LLM-driven question generation will support adaptive learning features and dataset augmentation for system testing.

3. Enhanced Context Retrieval for Complex Queries

For more nuanced academic questions, the system will explore **context expansion techniques** using LangChain's retriever framework and potentially Ollama-hosted LLMs. By dynamically retrieving additional supporting context, the platform can offer richer, more complete answers—even for open-ended or evidence-heavy queries.

4. Comprehensive Evaluation with RAGAS

Adopting the RAGAS suite for systematic evaluation will provide objective, fine-grained metrics for retrieval and generation quality, faithfulness, latency, and user satisfaction. This will enable more rigorous, data-driven system improvement.

5. Support for Additional Document Formats

Expanding the system to handle a broader variety of academic resources and file types, beyond PDFs, will increase its applicability across diverse curricula and research domains.

6. Enhanced PDF and Document Parsing

Future work will focus on supporting **complex academic PDF parsing** by integrating advanced tools such as LlamaParse and LangChain, inspired by recent methodologies. This will enable more reliable extraction of structured content, tables, formulas, and multi-column layouts from challenging documents, ensuring better semantic chunking and context retrieval.

7. Incorporation of User Feedback and Rich UI Enhancements

The system will continue to gather and incorporate direct feedback from students and educators, refining both the retrieval/generation pipeline and the user experience. Special attention will be paid to delivering a rich, interactive UI—potentially built with Streamlit/Angular/React.js, or other modern frameworks—offering intuitive document upload, query submission, answer visualization, and feedback options to ensure accessibility and engagement for all users.

8. Scalability and Institutional Deployment

Finally, the platform will be optimized for larger-scale deployment, supporting classrooms, labs, or entire institutions, with ongoing focus on data privacy, system robustness, and easy manageability.

These directions collectively aim to make the RAG pipeline more robust, user-friendly, and academically valuable, ensuring it remains adaptable as both the LLM landscape and educational needs continue to evolve.

Chapter 6
Bibliography / References

- Binita Saha, Utsha Saha et al. (Jan 2025) “QuIM-RAG: Advancing Retrieval-Augmented Generation with Inverted Question Matching for Enhanced QA Performance”, arXiv: 2501.02702

- M. Shah, P. Zhou, et al. (Oct. 2024), “A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions” (*ResearchGate*)

- Praban Narak (Apr. 2024), “RAG on Complex PDF using LlamaParse, Langchain and Groq”, Medim.com blog

- Rohan Anil, Sebastian Borgeaud, et al. (Dec 2023) “Gemini: A Family of Highly Capable Multimodal Models”, arXiv: 2312.11805

- Sonal Prabhune and Donald J. Berndt (Nov 2024), “Deploying Large Language Models with Retrieval Augmented Generation”, arXiv: 2411.11895v1

- Y. Liu, Y. Wang, et al. (Dec. 2023), “Retrieval-Augmented Generation for Large Language Models” arXiv: 2312.10997

- Bo Ni, Zheyuan Liu, et al.(Feb 2025), “Towards Trustworthy Retrieval Augmented Generation for Large Language Models: A Survey”, arXiv: 2502.06872

- Boci Peng, Yun Zhu et al. (Aug. 2024), “Graph Retrieval-Augmented Generation: A Survey”, arXiv: 2408.08921

- Alec M. (Mar. 2024), “Retrieval Augmented Generation with Groq API”, Groq blog

- Franklin Lee, Tengfei Ma (June 2025), “The Budget AI Researcher and the Power of RAG Chains”, arXiv: 2506.12317v1

- Jinhyuk Lee, Feiyang Chen et al. (Mar. 2025), “Generalizable Embeddings from Gemini”, Xiv: 2503.07891

- Petko Georgiev, Ving Ian Lei, et al. (Mar 2024)**,** “Gemini 1.5: Unlocking multimodal understanding across millions of tokens”, arXiv: 2403.05530

- Wenqi Fan, Yujuan Ding et al. (Aug. 2024) “A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models” ACM digital library: doi/10.1145/3637528.3671470

- Monica Riedler, Stefan Langer (Oct. 2024)**, “**Optimizing RAG with Multimodal Inputs for Industrial Applications”, arXiv: 2410.21943v1

Check list of items for the Final report

Is the Cover page in proper format?	Y
Is the Title page in proper format?	Y
Is the Certificate from the Supervisor in proper format? Has it been signed?	Y
Is Abstract included in the Report? Is it properly written?	Y
Does the Table of Contents page include chapter page numbers?	Y
Does the Report contain a summary of the literature survey?	Y
Are the Pages numbered properly?	Y
Are the Figures numbered properly?	Y
Are the Tables numbered properly?	Y
Are the Captions for the Figures and Tables proper?	Y
Are References/Bibliography given in the Report?	Y
Have the References been cited in the Report?	Y
Is the citation of References / Bibliography in proper format?	Y

Verified and Signed by Supervisor

Signed by Student

J. C. Chyudhury