Course Study Tool Using Retrieval-Augmented Generation

Christopher Varghese
cgvarghe@buffalo.edu
University at Buffalo
Buffalo, New York, USA

Vishnu Lokhande*
vishnulo@buffalo.edu
University at Buffalo
Buffalo, New York, USA

Abstract

This project presents a course study tool using a retrieval-augmented generation (RAG) system designed to assist graduate students in mastering concepts using PDF course materials. The system is built around three core tasks: (1) retrieving semantically relevant content from a collection of lecture slide PDFs using vector-based similarity search, (2) generating clear, detailed answers using a tuned Google FLAN-T5 large language model (LLM), and (3) enriching responses with relevant slide thumbnails retrieved from the original lecture material to provide visual context and support the RAG response. Using LangChain to orchestrate retrieval and structure the LLM inference pipeline, the system uses sentence-transformer embeddings and a FAISS (Facebook AI Similarity Search) vector store for efficient document indexing and similarity scoring. The extracted top two relevant slide images are rendered alongside responses in a full-stack interface, using Streamlit, to mimic a conversational tutoring environment. The RAG-based course study tool offers an offline, fully interactive study tool that grounds the generated explanations in the lecture content, improving clarity, retention, and accessibility for students.

CCS Concepts

• **Computing methodologies → Information extraction**; **Natural language generation**.

Keywords

RAG, LangChain, Hugging Face Transformers, Natural Language Generation, Semantic Search, PDF processing

ACM Reference Format:
Christopher Varghese and Vishnu Lokhande. 2025. Course Study Tool Using Retrieval-Augmented Generation. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym '25)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

With the growing volume of lecture materials and the increasing depth of knowledge required in graduate-level courses, students often encounter difficulties when reviewing slide decks to extract relevant explanations during self-study [8], [11]. Traditional search for keywords in PDFs [1] may be inadequate for learners who need a contextual understanding of complex concepts presented in lectures, such as STEM courses, where deep conceptual insight, stepwise reasoning, and multiple abstraction levels are critical [14].

In a typical office hours setting, the instruction-student dynamic emerges when concepts become hard to explain: students ask open-ended questions, and the instructor interprets the intent, retrieves the relevant slide or visual diagrams from lecture content, and reframes the explanation to match the learner's level of understanding. Recreating this multilevel dialogue at scale motivates our use of retrieval-augmented generation (RAG) [9]. By pairing semantic retrieval with an LLM [10], an RAG study tool identifies a student query, fetches supporting material, and generates a coherent and context-based answer similar to an instructor referencing content in real time [13].

To this end, we introduce a course-aligned study assistant built on the principles of RAG [15], designed to emulate personalized office hours support. The pipeline operates offline, supports student-driven queries, and cites original instructional content for transparency. Future work may explore multimodal understanding [4], adaptive feedback based on user learning profiles [3], and the deployment of multiple bots [2] tailored to smaller sections of lecture content.

We aim to complete the following tasks:

(1) **Semantic Slide Retrieval.** Process relevant content from a collection of lecture slide PDFs by embedding and indexing each document.
(2) **Natural Language Response Generation.** Accept a user query and return a clear, contextually grounded answer by prompting an LLM [10] and using a RAG approach [9].
(3) **Slide Visual Integration.** Enhance comprehension by attaching relevant slide thumbnails to the generated answer from the original lecture material.

2 Motivation for Slide-Based Visual Context

Lecture slides are one of the common formats used for instruction in graduate-level courses [5]. They combine structured text content with supporting visual elements—diagrams, formulas, tables—that are essential for understanding the material. By designing our system to operate directly on PDF slide decks, we create a generalizable architecture that can be used across various courses and disciplines. Once provided with slides in PDF format, the system can automatically extract content, generate semantic embeddings [?], and become a study assistant tailored to that course.

The semantic embeddings are dense vector representations using a sentence-transformer [12], all-MiniLM-L6-v2. This model

*Corresponding author.

has been trained on large-scale natural language inference and semantic textual similarity datasets, making it effective for encoding semantically similar text into proximate regions in vector space. By leveraging this pretrained transformer-based encoder, the system ensures that conceptually related chunks, such as those discussing the same deep learning algorithm or mathematical intuition, are closely aligned in the embedding space. During retrieval, user queries are encoded using the same model, and a similarity search identifies the most semantically aligned slide segments, ensuring that answers are not just keyword-matched but contextually and conceptually relevant to the user's intent.

This approach offers strong scalability and minimal instructor overhead: no additional annotation, API integration, or content restructuring is required.

In this paper, we evaluate the course study tool using material from CSE 676: Deep Learning, a graduate-level course at the University at Buffalo. The slide decks from this course present a challenge as they are text-dense, conceptually difficult, and contain mathematical formulations and various network diagrams. This pushes the system to handle both semantic retrieval of content and accurate grounding of responses in relevant visual context. Hence, this course offers an ideal environment to demonstrate the effectiveness and flexibility of our RAG system.

3 Related Works

Prior research has explored the use of retrieval-augmented generation and document-based question-answer systems for educational support.

In the educational domain, Jacobs and Jaschke [7] explored the integration of RAG with GPT-4 to provide contextual feedback on programming assignments. Their system used transcribed lecture recordings as an external knowledge base, enabling the generation of feedback that aligns closely with course materials and minimizes hallucinations. This approach underscores the potential of RAG in providing personalized educational support by focusing responses on specific instructional content.

Dong [6] proposed KG-RAG, a Knowledge Graph-enhanced RAG framework, which improves traditional RAG by integrating structured domain knowledge for more coherent and content-aligned responses. Their AI tutor system showcases the benefit of augmenting semantic retrieval with concept-linked graphs to reflect real-world educational dependencies between topics. This work is particularly relevant to the goal of our project of reinforcing the lecture context through semantic retrieval.

Swacha and Gracel [13] surveyed the use of RAG-based chatbots in education, analyzing 47 systems designed for various instructional contexts, from health sciences to computer science. Their study highlights the role of RAG in addressing hallucination problems in large language models while enhancing the relevance and reliability of content. They emphasize how RAG systems can effectively support independent learning and student engagement by enabling contextualized access to course materials.

Our work contributes to the growing landscape of educational RAG applications by combining efficient retrieval mechanisms, lightweight embeddings, and visual context to create a comprehensive study assistant tailored for graduate-level coursework.

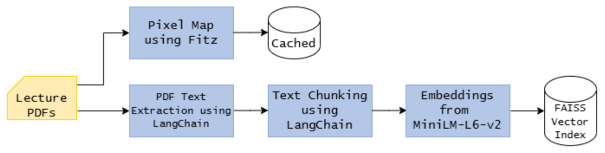
Pixel Map using Fitz
Cached
Lecture PDFs
PDF Text Extraction using LangChain
Text Chunking using LangChain
Embeddings from MiniLM-L6-v2
FAISS Vector Index

Figure 1: Illustration of the Slide Embedding Framework.

4 Semantic Slide Retrieval

In this section, we outline our approach for semantic slide retrieval. Specifically, we present the pipeline of processing the PDFs, which is used for visual slide integration and response generation.

4.1 Our Framework for Semantic Slide Retrieval

Figure 1 illustrates our embedding pipeline, which retrieves relevant content and formats the contextual input for the language model to generate responses to user queries.

4.2 PDF Embedding Pathway

We follow a standard document embedding pipeline for lecture slide ingestion and retrieval preparation. This pathway enables semantic indexing of PDF lecture slides for downstream retrieval-augmented generation.

Slide Thumbnail Extraction. To support slide visual retrieval, we use the fitz module (PyMuPDF) to extract images from PDF lecture slides. Specifically, the get_pixmap() function iterates over all pages of each slide deck and generates a pixmap per page. These pixmaps are converted into PNG byte arrays and cached in memory for fast lookup during chat interaction. These thumbnails are displayed alongside generated responses to show concepts visually.

Text Extraction and Chunking. For text processing, we use the PyPDFLoader from LangChain to extract structured content from each slide. The resulting pages are passed to the RecursiveCharacterTextSplitter, which divides the documents into overlapping textual chunks. We empirically select a chunk size of 400 and overlap of 50, which strikes a balance between preserving contextual continuity and respecting model input length constraints. Smaller chunk sizes improve retrieval granularity, while overlapping chunks help retain cohesion across the slide transitions.

Embedding and Indexing. Each chunk is embedded into a dense vector using the HuggingFaceEmbeddings wrapper over the MiniLM-L6 v2 model. This model is chosen for its strong semantic alignment performance and efficient CPU inference, enabling local deployment without GPU dependencies. The resulting embeddings are then stored in a FAISS (Facebook AI Similarity Search) index using FAISS.from_documents. This index supports fast approximate nearest-neighbor search during retrieval time.

The FAISS index stores dense vector embeddings and supports efficient top-k similarity search given a query vector (used in response generation). It returns the IDs and similarity scores of the most relevant matches, which can then be mapped back to their corresponding PDF slide chunks for contextual retrieval.

Persistence and Caching. The entire pipeline is cached using Streamlit's @st.cache_data and @st.cache_resource functions to

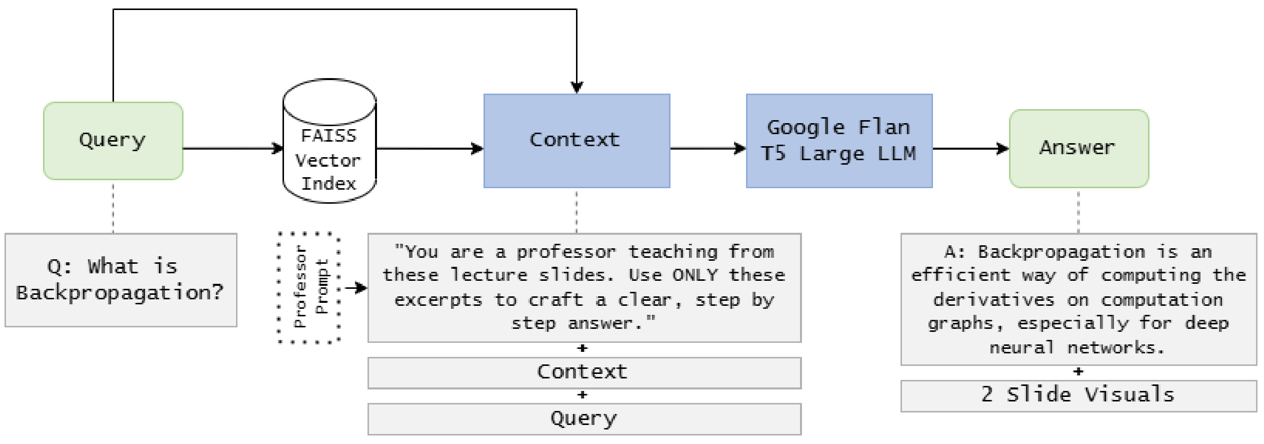
Query
FAISS Vector Index
Context
Google Flan T5 Large LLM
Answer
Q: What is Backpropagation?
Professor Prompt
"You are a professor teaching from these lecture slides. Use ONLY these excerpts to craft a clear, step by step answer."
+
Context
+
Query
A: Backpropagation is an efficient way of computing the derivatives on computation graphs, especially for deep neural networks.
+
2 Slide Visuals

Figure 2: Illustration of the Response Generation pipeline.

avoid redundant computation across sessions. If the index directory already exists, the FAISS vector store is loaded from disk; otherwise, it is built from the extracted documents and saved for future use. Users can manually trigger a rebuild of the embeddings via the sidebar interface, allowing for updates or reprocessing of newly added lecture materials.

This pipeline forms the foundation for semantic slide retrieval and embedding, supplying high-quality contextual representations to support accurate and relevant question answering in the next stages.

5 Natural Language Response Generation

In this section, we present our solution for response generation, outlining the pipeline that transforms a user query into a contextually rich answer using a large language model (Google Flan T5-Large).

5.1 Our Framework for Response Generation

Figure 2 illustrates our framework for response generation, which creates natural language responses grounded in the retrieved PDF context, from the FAISS index similarity search, from a local LLM.

5.2 LLM Response Generation Pipeline

We implement a retrieval-augmented generation (RAG) pipeline where an open-source large language model (LLM) is locally deployed to answer student queries based on lecture slides.

Model Setup. We use Hugging Face's FLAN-T5-large model via the transformers.pipeline API, configured for text-to-text generation. The model is deployed on the CPU to maintain local operability for students. Model parameters include max_length=512 to fit context within model limits, and sampling is enabled for more conversational output with temperature=0.9 to produce less robotic-sounding responses.

Prompt Construction. When a student submits a query, a similarity search is performed on the FAISS vector index to retrieve the top-8 relevant slide chunks. These chunks are formatted into a structured context prompt for the LLM, ensuring answers remain

Table 1: Libraries and Their Uses in the PDF Embedding Pipeline

Library / Module	Purpose in Pipeline
`fitz` (PyMuPDF)	Extracts slide thumbnails by rendering PDF pages into pixmap images for visual reference.
`PyPDFLoader` (LangChain)	Parses and extracts structured text from PDF slides for chunking and embedding.
`Recursive Character Text Splitter` (LangChain)	Splits the extracted text into overlapping chunks to maintain semantic continuity during embedding.
`HuggingFaceEmbeddings`	Converts text chunks into dense semantic vectors using the MiniLM-L6-v2 sentence transformer.
`FAISS`	Stores and indexes the embedding vectors for fast approximate nearest-neighbor search during retrieval.
`Streamlit`	Powers the full-stack web interface, including caching, session state, and interactive user input/output.

grounded in slide content, including text, references, and attributions retrieved from the slides.

Instructional Prompting. The context is combined with a fixed instruction prompt to guide the language model's tone and behavior.

"You are a university professor teaching from these lecture slides. Use ONLY these excerpts to craft a clear, step-by-step answer."

This helps keep the generation grounded in the CSE 676 slide content to avoid hallucinated content. The context and user query are appended as a direct question at the end of the prompt.

Answer Generation. The constructed prompt is passed to the local LLM pipeline via HuggingFacePipeline. The generated response is sent back to the user interface and cached in the session state for chat history retrieval.

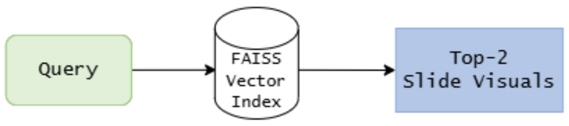

Figure 3: Illustration of the slide retrieval pathway.

5.3 Slide Visualization Retrieval

Slide Retrieval for Visual Support. The pipeline performs a similarity search retrieval on the LLM response and collects the top 2 most relevant slides. This is used to append to the RAG-generated response with contextual visual cues.

As a whole, the pipeline forms the core of our slide retrieval and visualization, enabling responsive, grounded, and context-aware question answering over a set of lecture materials.

5.4 Alternative Parallel Slide Retrieval Pipeline

An alternative pathway for slide visualization retrieval is shown in Figure 4. In this pathway, the top 2 slides are retrieved directly from the FAISS vector store using the original user query, in parallel with the context formation and LLM response generation.

While this approach simplifies the pipeline by decoupling slide retrieval from the generated answer, it proved less effective in practice. This is because the slides retrieved based on the query alone often lacked sufficient alignment with the final generated response. In particular, slides selected this way were sometimes visually relevant but missing the deeper explanation covered in the generated answer.

To address this, our primary implementation instead uses the final LLM-generated answer as the input to the FAISS vector store for the slide retrieval step. This ensures that the visual content returned is semantically aligned with the core concepts discussed in the answer. The query-only variant remains a viable option when parallel processing is prioritized, but is not used in our final system.

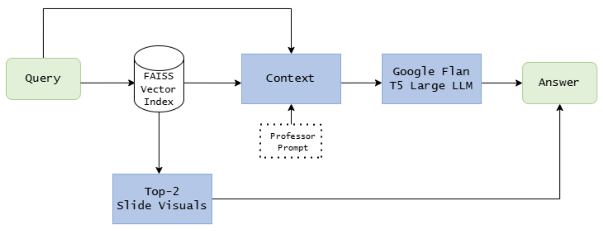

Figure 4: Illustration of alternative slide pathway.

6 Capabilities and Strengths

The Course Study RAG system is designed to function as a contextual study assistant for students. Below, we highlight the key capabilities that demonstrate the effectiveness of the system.

6.1 Contextualized Question Answering

The system excels at delivering answers that are grounded in the provided course material. Unlike generic chatbots or external LLMs that may diverge from course-specific context, the Course Study RAG maintains relevance by retrieving high-similarity slide content from the course PDFs before generation.

6.2 Visual Learning Support

By attaching the top 2 related slide thumbnails to each response, the system creates a visual supplement that can be an essential learning aid in education, where diagrams, graphs, and equations are central.

6.3 Flexible Interaction

The system supports open-ended, student-driven queries similar to the nature of office hours. Users can ask for explanations, comparisons, or specific visual slide references. The following diagram shows questions demonstrate the assistant's range and the kind of learning outcomes it supports within the CSE 676 Deep Learning course:

The RAG system produces responses conditioned on retrieved slide context and an instructional prompt. Because the model uses sampling with temperature, responses to the same query may differ across sessions. The outputs in Table 2 illustrate plausible output completions.

This set of capabilities positions the tool as an effective self-study resource for graduate-level deep learning students, especially in offline, asynchronous learning settings.

7 Conclusion

This paper presented a course-aligned study assistant that leverages Retrieval-Augmented Generation (RAG) to mimic the educational learning procedures using lecture slide PDFs. By integrating semantic retrieval with a tuned large language model, our system provides context-aware explanations and appends slide visuals that enhance interpretability.

We detailed a scalable pipeline that operates offline, supports student-driven queries, and emphasizes educational grounding by

Table 2: Sample Queries and RAG-Generated Answers

User Question	Example RAG Responses
Explain backpropagation using the example given in the lecture.	Backpropagation algorithm uses two passes through the computational graph. Forward propagation is for the computation of the sum of the outputs of each layer, while backward propagation is for the computation of the gradients of parameters in each layer, based on results from the forward pass.
What are the pros and cons of using ReLU over sigmoid?	Sigmoid saturates across most of the domain, saturates to 1 when z is very positive, and 0 when z is very negative. Saturation makes gradient learning difficult, e.g., difficult to set the step size for the whole range. ReLU and Softplus (smooth version of ReLU) increase (almost linearly) for input >0, which is easier for gradient learning.
Explain feed-forward networks with example visuals from the lecture.	Feed-forward networks provide a universal system for representing functions: Given a continuous function, there is a feed-forward network that approximates the function (given with 2 relevant slides from the feed-forward network lecture).

citing from original instructional content. The CSE 676 Deep Learning course was used as a case study to demonstrate the system's effectiveness under content-heavy and visually dense conditions.

Our findings suggest that this architecture, driven by document embeddings, contextual prompting, and visual augmentation, can serve as a generalizable solution across courses where self-study requires clarity. Future work may explore the incorporation of adaptive feedback based on user learning profiles and the deployment of multiple specialized bots tailored to lecture-specific content.

References

[1] H Riaz Ahamed and D Kerana Hanirex. 2024. Efficient E-Learning Multi-Keyword Search-Based Application for Students' Better Education. In *Cross-Industry AI Applications*. IGI Global, 319–336.

[2] Mia Allen, Usman Naeem, and Sukhpal Singh Gill. 2024. Q-module-bot: A generative ai-based question and answer bot for module teaching support. *IEEE Transactions on Education* (2024).

[3] Inés Alvarez-Icaza, José Molina-Espinosa, and Paloma Suárez-Brto. 2024. Adaptive Learning for Complex Thinking: A Systematic Review of Users' Profiling Strategies. *Journal of Social Studies Education Research* 15, 2 (2024), 251–272.

[4] Bettyjo Bouchey, Jill Castek, and John Thygeson. 2021. Multimodal learning. *Innovative learning environments in STEM higher education: Opportunities, challenges, and looking forward* (2021), 35–54.

[5] Jane Costello, Pam Phillips, Denise Carew, and Daph Crane. 2020. The Effective Use of Text, Visuals, and Audio in Online Graduate Learning. *The Finest Blend: Graduate Education in Canada* (2020), 103–140.

[6] Ying Dong. 2024. KG-RAG: Enhancing Retrieval-Augmented Generation with Knowledge Graphs for Education. *Applied Sciences* 15, 8 (2024), 4234.

[7] Blake Jacobs and Tim Jaschke. 2024. Generating Contextual Feedback for Student Code with RAG and GPT-4. *arXiv preprint arXiv:2311.17696* (2024).

[8] Firdavs Khaydarov, Haley R Peterson, and Carrie Lewis Miller. 2019. Assessing the significance of course materials for graduate program admission. *Smart Learning Environments* 6 (2019), 1–12.

[9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[10] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435* (2023).

[11] Kathrin F Stanger-Hall, Sarah Lang, and Martha Maas. 2010. Facilitating learning in large lecture classes: testing the "teaching team" approach to peer learning. *CBE—Life Sciences Education* 9, 4 (2010), 489–503.

[12] Lukas Stankevičius and Mantas Lukoševičius. 2024. Extracting sentence embeddings from pretrained transformer models. *Applied Sciences* 14, 19 (2024), 8887.

[13] Jakub Swacha and Wojciech Gracel. 2025. Educational Chatbots Based on Retrieval-Augmented Generation: A Systematic Literature Review. In *Advanced Learning Technologies (ICALT)*. Springer.

[14] Anastasia Olga (Olnancy) Tzirides, Gabriela Zapata, Nikoleta Polyxeni Kastania, Akash K. Saini, Vania Castro, Sakinah A. Ismael, Yu ling You, Tamara Afonso dos Santos, Duane Searsmith, Casey O'Brien, Bill Cope, and Mary Kalantzis. 2024. Combining human and artificial intelligence for enhanced AI literacy in higher education. *Computers and Education Open* 6 (2024), 100184. doi:10.1016/j.caeo.2024.100184

[15] Jiale Wei, Shuchi Wu, Ruochen Liu, Xiang Ying, Jingbo Shang, and Fangbo Tao. 2025. Tuning LLMs by RAG Principles: Towards LLM-native Memory. *arXiv preprint arXiv:2503.16071* (2025).