MACHINE ASSISTANT WITH RELIABLE KNOWLEDGE: ENHANCING STUDENT LEARNING VIA RAG-BASED RETRIEVAL

Yongsheng Lian
EduQuery AI
Louisville, KY 40223
eduqueryai@gmail.com

ABSTRACT

We present **Machine Assistant with Reliable Knowledge (MARK)**, a retrieval-augmented question-answering system designed to support student learning through accurate and contextually grounded responses. The system is built on a retrieval-augmented generation (RAG) framework, which integrates a curated knowledge base to ensure factual consistency. To enhance retrieval effectiveness across diverse question types, we implement a hybrid search strategy that combines dense vector similarity with sparse keyword-based retrieval. This dual-retrieval mechanism improves robustness for both general and domain-specific queries. The system includes a feedback loop in which students can rate responses and instructors can review and revise them. Instructor corrections are incorporated into the retrieval corpus, enabling adaptive refinement over time. The system was deployed in a classroom setting as a substitute for traditional office hours, where it successfully addressed a broad range of student queries. It was also used to provide technical support by integrating with a customer-specific knowledge base, demonstrating its ability to handle routine, context-sensitive tasks in applied domains. **MARK is publicly accessible at app.eduquery.ai**.

1 Introduction

College education is facing significant challenges as increasing numbers of students arrive underprepared, particularly in foundational subjects like mathematics. Recent studies show that many university students struggle with basic mathematical concepts and lack the skills necessary for college-level coursework [1]. According to the National Assessment of Educational Progress (NAEP), nearly 40% of high school graduates demonstrate insufficient proficiency in mathematics [2, 3]. This academic gap hinders their performance in college and contributes to broader issues of retention and achievement. While one-on-one tutoring has proven effective, it is logistically impractical in large-enrollment courses. The scalability of traditional office hours is further constrained by scheduling conflicts among faculty, teaching assistants, and students—many of whom juggle academic responsibilities with part-time employment. These limitations reduce access to timely, personalized support when it is most needed.

Traditional question-answering (Q&A) systems, such as expert systems, are capable of addressing student inquiries by retrieving the most relevant text excerpts from a predefined corpus. Because these responses are drawn from verified, educator-approved materials, they tend to be accurate and trustworthy [4, 5]. These systems often emphasize step-by-step problem-solving but lack the capacity to interpret or explain the underlying concepts behind a question. Their performance is highly dependent on the quality, structure, and coverage of the underlying knowledge base. Moreover, they are typically not well aligned with evolving curriculum objectives or the individualized needs of learners [6].

Large language models (LLMs) have achieved impressive results across a wide spectrum of natural language tasks [7, 8, 9, 10, 11]. They achieve high scores on a wide range of college-level benchmarks like AP, GRE and SAT tests [12, 13]. LLMs even reach the highest performance on challenging Olympiad math [14, 15]. LLMs can be used in higher education. Most recent studies on the use of LLMs in higher education focus on high-level discussions of ethics, policy, and general applications, rather than exploring subject-specific implementations [16, 17, 18]. While these studies acknowledge the potential of AI to enhance learning, they also highlight risks such as academic misconduct, bias, and misinformation, which can hinder the development of critical thinking skills. However, there is a limited research focused on integrating AI into specific courses—largely due to the scarcity of conversational educational data [19].

1.1 Hallucination from LLMs

Despite their remarkable capabilities, LLMs suffer from a critical limitation: hallucination-the generation of outputs that are coherent and plausible but factually incorrect or misleading [20, 21]. These inaccuracies are particularly problematic in educational contexts, where students may unknowingly adopt false information, leading to flawed conceptual understanding and persistent misconceptions [22]. The risk is amplified when AI-generated content is integrated into curricula or assessments without proper verification [23].

This challenge becomes even more pronounced in specialized domains such as aerospace engineering. For instance, a recent study evaluating LLMs in aerospace manufacturing revealed that the models frequently produced incorrect or misleading answers to domain-specific queries, posing a risk to students developing accurate mental models of critical technical concepts [24].

Furthermore, students with limited subject matter expertise are especially vulnerable to overreliance on AI-generated responses. Without the skills to independently verify information, they may accept erroneous outputs at face value, compounding the problem of misinformation in learning environments [25].

These hallucinations inherent from the fundamental architectural and training limitations. First, LLMs are trained to predict the next word/token, not to verify the fact [20]. They may generate plausible but incorrect information that fits learned patterns. Second, the parametric nature of LLMs imposes an upper bound on their knowledge capacity [26]. OpenAI's GPT-4 is reportedly to have 1.8 Trillions of parameters, much larger than 85 billions of neurons in our brain. However, no model can fully encode all human knowledge, especially for rare or domain-specific facts. Third, LLMs are trained on static data, meaning their knowledge becomes outdated over time [12]. This issue is particularly pronounced in domains with highly specialized or long-tail content, such as flight dynamics or fluid mechanics, where domain-specific data may be scarce or poorly represented in public datasets. As a result, LLMs may struggle to provide accurate and contextually relevant responses in these areas. Last, LLMs are trained on vast internet-scale corpora, which contain factual inaccuracies, contradictions, and misinformation [27]. For example, when asking LLMs to explain lift on a wing, they still generated answers that the velocity on the top surface is higher than the bottom surface because air needs to reach the trailing edge at the same time even though the answer is incorrect and has been corrected by professors in the classroom.

1.2 LLMs+RAG

Retrieval-Augmented Generation (RAG) has emerged as a promising approach to mitigate some of the challenges faced by LLMs, particularly hallucination and limited knowledge generalization [28, 29, 30, 31]. RAG is an architecture designed to enhance LLMs by integrating external knowledge retrieval into the text generation process. The external knowledge typically comes from verified and curated sources, such as Wikipedia, technical manuals, textbooks, lecture notes, or proprietary knowledge bases. Since the retriever accesses content at inference time, RAG models can ground outputs in reliable documents, ensuring responses are both relevant and verifiable. Studies have shown that RAG-based systems can produce more factually accurate and specific outputs across a variety of tasks [27, 32]. For instance, Lewis et al. [28] demonstrated that RAG significantly outperformed parametric-only models on several open-domain Q&A benchmarks, while also enabling models to cite evidence and improve transparency. Studies have shown that RAG offer a balance between adaptive responses and curriculum alignment [33].

Additionally RAG facilitates knowledge freshness by retrieving from up-to-date corpora during inference. This capability enables LLMs to adapt to the most updated knowledge without costly retraining. RAG can also handle specialized tasks by searching custom sources like scientific papers without costly retraining of LLMs.

Several universities have successfully integrated LLM+RAG systems into college courses, demonstrating their efficiency and effectiveness in supporting student learning. At the University of Pennsylvania, ChemTAsk—a RAG-enabled open-source tool—was deployed in an advanced biological chemistry course to answer student questions using lecture notes and primary literature. The study showed ChemTAsk matched the performance of human teaching assistants. Students rated it as accurate, helpful, and faster than TAs [34]. Similarly, the University of Michigan's AI-University (AI-U) system was used in a graduate-level finite element methods course to deliver instructor-aligned responses by leveraging lecture videos, notes, and textbooks. The system demonstrated strong alignment with course materials [35]. The University of California, Irvine used RAGMan, an AI-powered tutoring system designed for an introductory programming course [36]. RAGMan utilizes RAG to provide students with homework and general programming assistance without giving direct answers. The system demonstrated high accuracy for questions within its intended scope. Student feedback indicated that the AI tutors were helpful for learning, with 78% of users reporting improved learning outcomes [36]. Feng et al. [37] has used LLM-based tutoring system tailored for computer science education. Student feedback showed that the system increased the accessibility of course-specific tutoring help and shortened the feedback loop.

While LLM+RAG systems have shown promise in enhancing college-level instruction, outstanding challenges remain to limit their broader effectiveness and scalability. First, studies have shown that the quality of user input material directly affects the accuracy and relevance of generated responses [34, 35]. Without curated, comprehensive and unbiased content, these systems risk producing incomplete or misleading answers. Second, students may over-reliance on AI tools, potentially reducing engagement with core learning activities such as critical reading, discussion, or problem-solving [36]. Third, hallucinated responses remain a challenge, particularly in open-ended or ambiguous queries [38]. Last, many systems require substantial instructor effort to configure and align the AI responses with course pedagogy, which can be a barrier to adoption, especially in resource-limited settings.

We present MARK (Machine Assistant with Reliable Knowledge), a general-purpose LLM+RAG system designed to support both educational and technical domains by delivering accurate, contextually grounded responses. MARK is built on a RAG framework. To optimize retrieval across diverse query types, MARK employs a hybrid search mechanism that integrates both sparse and dense retrieval methods. A built-in feedback system enables students to rate responses and allows instructors to review and revise outputs. Beyond education, MARK can also be deployed for technical support.

2 Retrieval Methods

An effective RAG system relies critically on the quality of its passage retrieval component, which identifies and selects the most relevant context passages from a large corpus to support accurate response generation. To meet the demands of diverse and often complex user queries, the retrieval method must not only be computationally efficient, but also semantically precise, ensuring that the retrieved passages are both relevant and contextually appropriate.

2.1 Keyword-Based Search

Keyword-based search is a simple and widely used approach for information retrieval. With this search, documents are selected based on the presence of exact query terms. This method relies on direct matching of query keywords against document content, often using Boolean logic or exact string matching. While fast and interpretable, keyword search does not account for term importance or semantic meaning, making it less effective in handling synonyms, paraphrases, or ambiguous queries [39].

2.2 Sparse Vector Space Models

Sparse vector space models represent documents and queries as high-dimensional vectors in which most elements are zero. They capture only the presence and frequency of specific words. Approaches such as Term Frequency–Inverse Document Frequency (TF-IDF) [40] and BM25 [41] exemplify this class of methods.

Like keyword-based methods, sparse vector space models rely on exact term matching. These models also rely on statistical weighting such as term frequency and inverse document frequency. These models assign higher weighting factors to terms that are more effective at differentiating between documents, which improves the accuracy of document ranking in response to a query. Despite their simplicity, these models remain popular due to their efficiency and strong performance in low-resource or domain-specific retrieval scenarios.

In these models, a collection of N documents is represented as $\mathcal{D} = \{d_i\}_{i=1}^{N}$, and the vocabulary consists of M unique terms as $\{t_j\}_{j=1}^{M}$. Each document d_i is represented by a vector $\vec{d_i} \in \mathbb{R}^M$, where the j-th component of the vector is the weight of term t_j in document d_i.

2.2.1 TF-IDF

In the TF-IDF model, the weigh of term t_j in a document d_i is computed based on term frequency (TF), which measures how often t_j appears in d_i, and inverse document frequency (IDF), which measures how rare t_j is acorss the entire document collection. The TF is defined as follows:

$$\text{TF}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}, \tag{1}$$

where $f_{t,d}$ is the number of times term t appears in document d. The more frequently a term appears in a document compared to other terms, the higher its TF value.

The IDF is defined as follows:

$$\text{IDF}(t, D) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right), \tag{2}$$

where N is the total number of documents and $|\{d \in D : t \in d\}|$ is the number of documents containing term t. The less frequently a term appears across all documents in the corpus, the higher its IDF value. This reflects the idea that rare terms are more informative and carry greater discriminative power when identifying the content of a specific document.

Finally the TF-IDF Weight is calculated as the product of TF and IDF values as follows:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \cdot \text{IDF}(t, D) \tag{3}$$

With TF-IDF, the whole corpus is represented by a matrix of shape $N \times M$.

To identify the document that best matches a user query using TF-IDF, the query is first represented as a TF-IDF vector. The TFs are computed based on the terms represent in the query, while the IDFs are derived from the full document corpus D. This ensures the query representation is consistent with the document vectors, allowing for accurate similarity comparisons.

Next, we compute the cosine similarity between the query vector $\vec{q}$ and each document vector $\vec{d_i}$ in the corpus. Cosine similarity measures the angle between two vectors in high-dimensional space, capturing how closely aligned they are in terms of content:

$$\text{similarity}(\vec{q}, \vec{d_i}) = \frac{\vec{q} \cdot \vec{d_i}}{\|\vec{q}\| \cdot \|\vec{d_i}\|} \tag{4}$$

Documents with the highest similarity scores are considered the most relevant and are selected as the retrieval results.

2.2.2 Best Match 25(BM25)

BM25 is a ranking function used in information retrieval to estimate the relevance of documents to a given query [41]. It is based on a bag-of-words model and extends the TF-IDF approach discussed above with more robust term frequency saturation and document length normalization.

For a multi-word query $Q = \{q_1, q_2, ..., q_m\}$, the BM25 score of document d_j is:

$$\text{BM25}(Q, d_j) = \sum_{i=1}^{m} \text{IDF}(q_i) \frac{f(q_i, d_j)(k_1 + 1)}{f(q_i, d_j) + k_1 \left(1 - b + b\frac{|d_j|}{\text{avgdl}}\right)} \tag{5}$$

Where:

- q_i: each term/word in the query.

- $f(q_i, d_j)$: frequency of term q_i in document d_j, i.e., the raw count of how many times q_i appears in document d_j.

- $|d_j|$: length of document d_j

- avgdl: average document length in the corpus

- k_1: term frequency scaling constant (typically 1.5)

- b: length normalization constant (typically 0.75)

The IDF used in BM25 is different from that used in TF-IDF (Eq. 3). The formula uses a log function that reduces the impact of very rare terms and stabilize scoring. The BM25 IDF is calculated as:

$$\text{IDF}(q_i) = \log\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right) \tag{6}$$

Where: - N: total number of documents. - $n(q_i)$: number of documents that contain term q_i.

2.3 Dense Vector Search

Dense vector search, also known as semantic retrieval, has emerged as a foundational technique in modern information retrieval systems. It represents queries and documents as low-dimensional, dense vectors using deep learning models such as BERT or other transformer-based encoders [42]. Unlike sparse methods or keyword-based method that rely on exact word matches, dense retrieval captures the semantic meaning of text, enabling it to retrieve relevant passages even when there is no word match with the query [43]. This paradigm is particularly effective for tasks requiring semantic matching, such as open-domain question answering, recommendation systems, and large-scale similarity search.

In dense vector search each document $d_i \in \mathcal{D}$ is mapped to a dense vector $\vec{v}_i \in \mathcal{R}^d$ via an embedding function:

$$\vec{v}_i = f_d(d_i; \theta), \tag{7}$$

where $\mathcal{D} = \{d_i\}_{i=1}^N$ denotes a corpus of documents, $f_d : \mathcal{X} \to \mathcal{R}^d$ is typically parameterized by a deep neural network, and θ denotes the learnable parameters.

Given a query $q \in \mathcal{X}$, its vector representation is computed as:

$$\vec{v}_q = f_q(q; \theta) \tag{8}$$

The retrieval task is similar to the TF-IDF by compute the similarity between the query vector $\vec{v_q}$ and the dense vector $\vec{v_i}$ as shown in Eq.4.

Previous studies have shown when evaluated on a wide range of open-domain Q&A datasets, dense retriever outperforms the BM25 system [43]. Recent advancements have made dense retrieval more practical and powerful, especially when combined with approximate nearest neighbor (ANN) techniques like FAISS [44].

The ANN makes dense retrieval more efficient. In ANN, the retrieval process seeks

$$\arg\max_{\vec{v}_i \in \mathcal{V}} \text{sim}(\vec{v}_q, \vec{v}_i), \tag{9}$$

where $\mathcal{V} = \{\vec{v}_i = f(d_i; \theta)\}_{i=1}^N$. However, exact nearest neighbor search in high-dimensional spaces is computationally expensive, especially when N is large. ANN algorithms address this challenge by trading a small amount of accuracy for a substantial gain in speed and scalability.

Several ANN techniques have become standard in dense retrieval system. Hierarchical Navigable Small World (HNSW) graphs (Malkov & Yashunin, 2018) constructs a multi-layer graph structure that allows logarithmic-time search over the dataset. It provides high recall and fast retrieval even in large-scale applications. Product Quantization (PQ) [45] compresses vectors by dividing them into subvectors and quantizing each subvector independently. This enables efficient distance computation in compressed space and dramatically reduces memory footprint.

The retrieval efficiency can be further improved by using FAISS (Facebook AI Similarity Search) [44]. FAISS is a widely adopted ANN library that supports various indexing strategies, including HNSW, PQ, and inverted file systems. It is optimized for both CPU and GPU execution, making it suitable for real-time, large-scale retrieval tasks.

Given a collection of N dense vectors $\{\vec{v}_i \in \mathbb{R}^d\}_{i=1}^N$, FAISS supports construction of various index structures $\mathcal{I}$ to efficiently solve the problem:

$$\vec{v}_{\text{NN}} = \arg\max_{\vec{v}_i \in \mathcal{V}} \text{sim}(\vec{v}_q, \vec{v}_i), \tag{10}$$

for a query vector $\vec{v}_q \in \mathbb{R}^d$, using similarity functions such as inner product or cosine similarity. In our system, we adopt FAISS to enable fast and accurate hybrid retrieval.

2.4 Hybrid search

Hybrid search leverages the complementary strengths of sparse and dense retrieval methods to enhance information retrieval, particularly in open-domain search and question-answering tasks. Sparse retrieval techniques, such as BM25 and TF-IDF, are effective at identifying exact term matches and highlighting rare but important keywords. In contrast, dense retrieval methods, which use neural embeddings, excel at capturing semantic similarity even in the absence of lexical overlap. By integrating both approaches—either through late fusion (e.g., combining retrieval scores) or unified indexing strategies—hybrid search systems can achieve higher recall and precision. Empirical studies demonstrate that hybrid retrieval consistently outperforms either method alone across a range of benchmarks [46, 47].

2.5 Re-Ranking

In a RAG framework, re-ranking serves as a crucial intermediary between initial document retrieval and final answer generation by a LLM. The process begins by independently encoding the query q and each document $d_i \in \mathcal{D}$ into vector representations using a shared or separate embedding function. These representations are then compared using a similarity function such as cosine similarity:

$$\text{sim}_{\text{retr}}(q, d_i) = \cos\left(f_q(q), f_d(d_i)\right), \tag{11}$$

where f_q and f_d map the query and docment to their respective vector embeddings. When using the same encoder for both, the embeddings lie in a shared semantic space, improving alignment and enabling fast approximate retrieval. However, this bi-encoder setup encodes the query and document independently, which limits its ability to capture fine-grained semantic or syntactic interactions between the two.

To improve the quality of retrieved contexts, a re-ranking stage is applied using a more powerful cross-encoder model $s(q, d_i)$, which takes the query–passage pair as joint input:

$$s(q, d_i) = f_{\text{cross}}([q; d_i]), \tag{12}$$

where $[q; d_i]$ denotes the concatenation of the query q and passage d_i, and f_{cross} is a neural scoring model (e.g., BERTA [42], RoBERTa [48] or SBERT [49]) trained to predict relevance or entailment.

The re-ranked list $\{d'_1, d'_2, \ldots, d'_k\}$ is obtained by sorting the original candidates based on the cross-encoder scores $s(q, d_i)$:

$$\text{ReRank}(q, \{d_i\}) = \text{argsort}_{i=1}^{k}\ s(q, d_i). \tag{13}$$

These top-ranked passages are then passed to the generator, which uses them to produce the final response. Re-ranking is especially effective in domains where precision and grounding are critical, as it prioritizes passages that are not just semantically similar but also contextually and pedagogically aligned with the user's intent.

3 Frontend Development

This section describes the frontend architecture of the system, which manages all user interactions. The frontend is built using HTML and JavaScript, providing a responsive, accessible, and platform-independent user experience. It includes two main interfaces: a student interface, which allows users to submit questions, receive AI-generated responses, and provide feedback; and an instructor interface, which enables faculty to monitor student interactions, review and revise AI responses, and contribute additional curated content. The entire web interface is deployed on Google Cloud Platform (GCP) and delivered securely through Cloudflare, ensuring global availability, low latency, and protection against DDoS attacks.

3.1 Student Interface

The student interface allows students or customers to submit natural language queries. As shown in Figure 1, this interface includes a customizable header—featuring an icon and styling options that instructors or administrators can configure.

Users enter their questions into a text input field and submit them using a designated button. Upon submission, the frontend issues a secure API request to the backend service. The backend processes the query, performs retrieval and generation operations, and returns a response via an encrypted connection. The frontend then renders the result within the interface, providing a seamless question-answering experience.

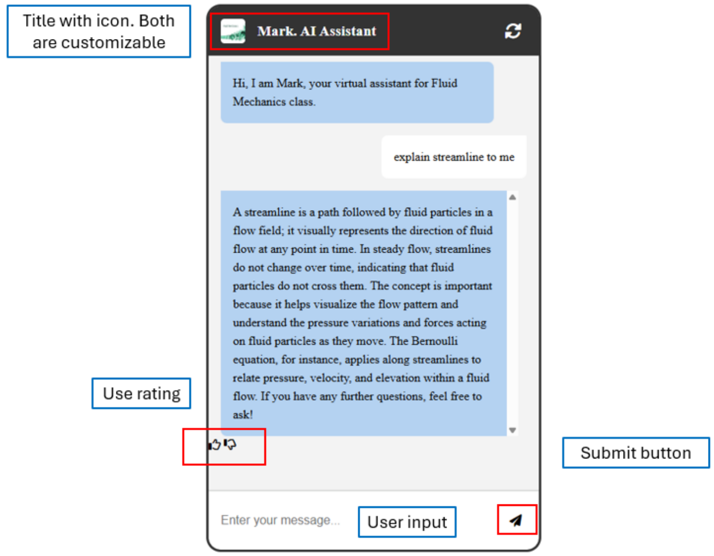

Figure 1: Interface of MARK with Text Entry and User Feedback

3.2 Instructor Interface

The instructor interface is designed for instructors and IT support personnel to manage knowledge base content, oversee chatbot behavior, and provide corrective feedback. This interface offers four core functionalities: creating, training, testing, and monitoring chatbots, which are shown in Figure 2.

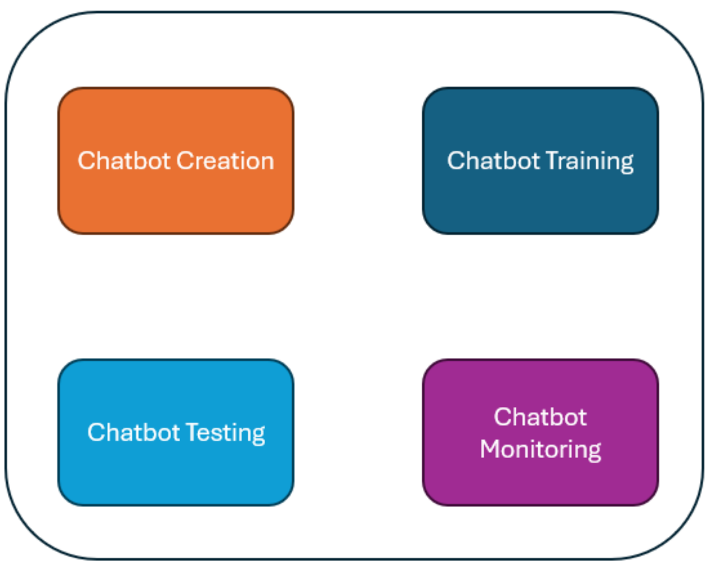

Figure 2: The four frontend modules of the instructor interface

Chatbot Creation Panel Figure 3 displays the interface for initializing a new chatbot. In this view, users can assign a unique name to the chatbot and specify a custom greeting message to be presented during initial interactions. An additional configuration option allows users to control the degree to which the underlying LLM can incorporate external knowledge beyond the RAG database. This is represented as a tunable parameter: a value of 0 strictly limits the LLM to information retrieved from the RAG database, while a value of 100 allows unrestricted use of external knowledge sources by the LLM.

Once configured, the chatbot can be deployed either as a standalone HTML page or seamlessly embedded within an existing website, such as a popup widget or inline element.

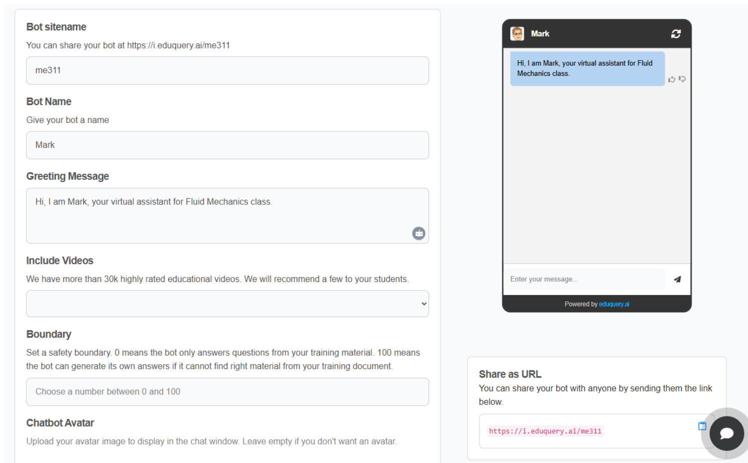

Figure 3: Interface of Create a New Chatbot

Chatbot Training Panel Figure 4 shows the panel to upload documents into the RAG database. This panel supports uploading files in PDF, CSV, or plain text formats. For users who wish to incorporate content from a webpage, we recommend first converting the webpage into a PDF file to ensure consistent formatting and easier processing. Once uploaded, the documents undergo a series of preprocessing steps—such as text extraction, cleaning, and formatting—to ensure they are properly indexed for retrieval. The detailed procedure for integrating these documents into the RAG database will be addressed in the backend development section.

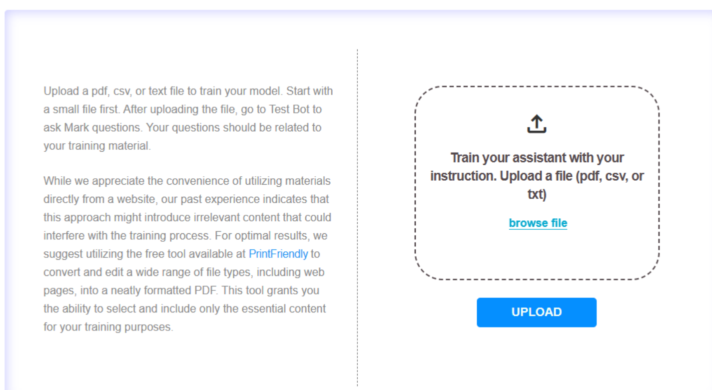

Figure 4: Interface of Uploading Contents for RAG

Chatbot Testing Panel Following content ingestion, the chatbot can be evaluated through the Testing Panel, which is designed to simulate real-world user interactions. This interface mirrors the layout and functionality of the student interface UI shown in Figure 1. This is to ensure consistency across testing and deployment environments. The Testing Panel enables instructors or system administrators to validate the chatbot's responses against expected outputs, assess performance accuracy, and identify potential failure cases prior to public deployment.

Chatbot Monitoring Panel Instructors can review the chatbot's responses and provide corrective feedback through the Monitoring Panel, as illustrated in Figure 5. The panel allows users to filter and select chatbot responses generated within a specified time range. Revised or improved responses can be entered into the correction input area on the right.

Upon clicking the Save button, the updated content is immediately inserted into the RAG database, making it available for future inference queries.

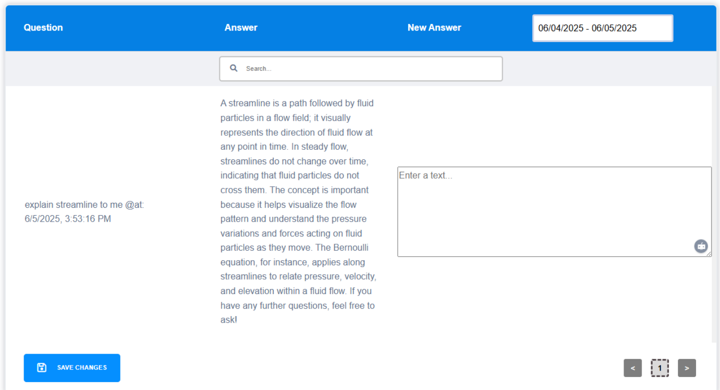

Figure 5: Interface of Monitoring AI Responses

4 Backend Development

The backend architecture of MARK is implemented using the Python Flask web framework. Flask is a lightweight and efficient foundation for handling HTTP requests and serving the application's API endpoints. This backend is responsible for orchestrating tasks such as chatbot training, RAG database updates, and response generation. The Flask server is deployed on Google Cloud. The four key modules in the backend are shown in Figure 6.

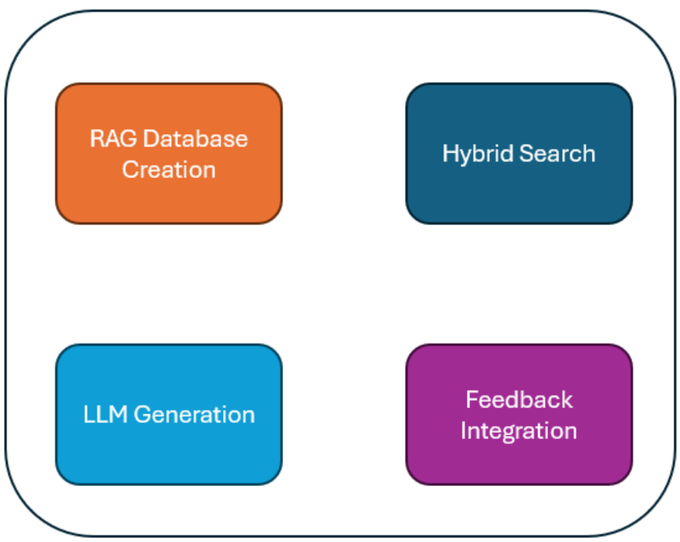

Figure 6: The four backend modules

4.1 RAG Database Preparation

The uploaded documents from the frontend are processed through a backend API to prepare them for embedding and retrieval. For plain text and comma-separated values files, the content is segmented directly into smaller chunks. For PDF files, a preprocessing step is required to extract and convert the content into structured textual data. We developed a Python-based pipeline for this task using the PyMuPDF library, which allows direct access to each page and its associated text content.

The extraction process iterates over each page of the document, retrieving raw text data and optionally applying standard preprocessing techniques such as line break removal and whitespace normalization. The resulting cleaned text is then divided into semantically coherent chunks. Based on our experiments, including the section heading within each chunk enhances retrieval performance by providing additional context.

Chunk length should be tailored to the expected length of the response; shorter responses benefit from smaller chunks. For chatbot application, the answer length is usually short. Hence, we used a maximum chunk length of 384 tokens.

Each text chunk is subsequently embedded into dense vector representation, as described in Eq. 7. In general, higher-dimensional embeddings can capture more nuanced semantic information by providing a larger representational space. This is especially beneficial in large-scale retrieval systems where fine-grained distinctions between semantically similar documents are critical. However, for smaller corpora, the marginal gains from increasing the embedding dimension beyond a certain point tend to diminish, while the computational and memory costs grow linearly with dimensionality.

Our empirical results indicate that a 512-dimensional vector offers a good balance between performance and efficiency for our use case. In our application, we employed the 'jina-embeddings-v2-small-en' model [50], a monolingual English model with a maximum input length of 8192 tokens. Based on a BERT-like architecture, this model is pretrained with sequences of up to 512 tokens but can generalize to longer sequences. The embedding tool map each chunk into a 512-dimension dense vector. With only 33 million parameters, it offers highly efficient and fast inference while maintaining competitive performance for RAG applications. For multi-language embedding, jina3 [51] can be used.

4.2 Hybrid RAG Retrieval

When a user query is submitted, the backend initiates a hybrid retrieval process that combines both sparse and dense search paradigms to maximize relevance and coverage. For the sparse retrieval component, we employ the BM25 algorithm (see Section 2.2.2). BM25 can identify documents containing exact keyword matches.

In parallel, dense retrieval is performed using vector similarity search powered by FAISS. Each text chunk in the RAG database is pre-encoded into a fixed-size dense embedding vector using the Jina Embeddings v2 model. The query is embedded into the same vector space, and FAISS retrieves the top-ranked chunks based on vector similarity (see Equation 4). We then merge the top results from the BM25 and FAISS queries into a unified context document. This combined context is then re-ranked and the top k passages are forwarded to LLM for final response generation.

4.3 LLM and its Prompt

The retrieved document is subsequently passed to a LLM to generate a response. For this purpose, we utilize the OpenAI API. To ensure accurate and contextually relevant answers, we provide a carefully designed prompt that instructs the LLM to ground its responses strictly in the retrieved content. The prompt reads as follows:

MARK Prompt

> You are a knowledgeable and patient teaching assistant. Refer to the previous conversation when answering, and politely decline if a question is outside your knowledge. Be concise.

This prompt encourages the model to adopt a helpful and precise tone, while constraining its output to information supported by the retrieved document, thereby reducing the likelihood of hallucinations and improving factual consistency.

4.4 Integrate Feedback into RAG Database

An essential component of the MARK is its feedback integration mechanism. It enables users to review and refine the AI-generated responses. Corrected or improved answers can be submitted directly through the instructor interface. Once received, this feedback is incorporated into the existing RAG knowledge base. On the backend, the new text entries are first appended to the raw document repository. Subsequently, both the sparse index and the dense vector index are updated to reflect the new content. For small to moderately sized datasets—fewer than one million entries—index regeneration is computationally efficient. In our deployment, these updates are executed on a Google Cloud instance equipped with 4 Intel Cascade Lake CPUs and 32 GB of RAM.

5 Results

5.1 MARK as Virtual Tutor

We developed a Fluid Mechanics module using MARK for Fluid Mechanics class. Fluid Mechanics is a required course in Mechanical Engineering field. It covers fundamental topics like hydrostatics, Bernoulli's equation, control volume analysis, dimensional analysis, and internal and external flows. We created the basic RAG database using class syllabus, class notes, and homework problems. The database is further enhanced with curated answers to student questions by instructors. The experiment was conduced in the fall of 2023.

Students accessed MARK via a dedicated link provided by the instructor. Students can submit questions in natural language at any time. Students often asked MARK about administrative and logistical aspects of the course, such as exam schedules, the scope of material covered in upcoming exams, homework deadlines, and clarification of class policies. Although these details were clearly documented in the syllabus, the frequency of such queries suggests that students preferred a quick, centralized, and easily accessible source of answers over traditional course documentation.

In addition to logistical questions, students also posed academic queries, particularly related to homework assignments. These questions were typically brief, such as "How to solve Homework 5.6," reflecting a desire for direct assistance. MARK responded with complete solutions rather than offering step-by-step guidance. This pattern suggests the potential value of refining MARK's instructional strategies through example prompts or usage training to encourage deeper learning.

Students also posed playful questions such as 'Mark, who is your dad?', reflecting the natural curiosity that often accompanies the early adoption of emerging technologies like generative AI. These interactions suggest that students were not only testing the bot's capabilities, but also relating to it as a conversational partner—much like how they might engage with a personality on a social media app.

Student inputs to MARK are limited to text only, without the ability to upload diagrams, equations, or their intermediate work. This limits MARK's ability to assess student understanding or to provide feedback tailored to their specific problem-solving approach. Future iterations may benefit from incorporating multimodal input or answer checking features to better support formative assessment.

Overall adoption during the 2023 pilot phase was limited, mirroring national hesitations and skepticism following the widespread attention sparked by the release of ChatGPT-3.5 in November 2022. This hesitation was further compounded by the absence of clear institutional policies or guidance on the appropriate use of AI tools in the classroom.

5.2 MARK as Technical Support

MARK can be tailored to function as an intelligent technical support assistant, designed to address a wide range of customer inquiries. Unlike traditional FAQ pages, which require users to manually scan through lengthy and often irrelevant lists of answers, or rule-based chatbots that rely on rigid decision trees and canned responses, MARK delivers dynamic, context-aware support. It interprets user queries in natural language, retrieves relevant knowledge from an external document corpus, and generates human-like responses in real time. This allows MARK to handle a far broader and more nuanced range of questions than systems constrained by static templates or pre-scripted flows. By offering immediate, 24/7 assistance tailored to individual customer needs, MARK significantly improves the efficiency and effectiveness of technical support while reducing user frustration and support latency.

A key advantage of MARK is its ability to draw on vast knowledge bases—including product documentation, troubleshooting guides, and historical support records—to generate accurate, context-aware responses in real time. Since many customer queries are repetitive or follow recognizable patterns, the system can retrieve similar past cases or relevant technical documents and synthesize them into coherent, actionable responses using its generative capabilities. This allows MARK to deliver more helpful and comprehensive answers than traditional rule-based systems, while continuously improving as new data becomes available.

We show a few examples to demonstrate what MARK can do in technical support and customer service.

Give Instructions

Customer: I need help setup my wifi router

MARK: To set up your wifi router, visit m.******.com. If you need help, email us at contact@****.com.

Customer: I would like to change my plan

MARK: To change your plan, please log in to your account and navigate to the 'Add Data' page. From there, you can select a new plan. Any remaining data from your current plan will roll over, ensuring you don't lose any unused data. If you encounter any issues, feel free to ask for assistance!

Troubleshoot

Customer: my wifi hotspot does not work.

MARK: If your device does not work, please follow these troubleshooting steps:
1. Ensure that your SIM card is activated.
2. Change your device's Access Point Name (APN) settings.
3. Restart your device after making the APN change.
4. Confirm that your device is an unlocked 4G device.
If you still encounter issues, email us at contact@****.com for further assistance.

MARK can be integrated with internal databases and customer relationship management systems, which enables MARK to provide highly personalized assistance, such as verifying customer email addresses, checking account balances, or diagnosing issues with a specific device registered to a particular user. This multi-source integration enables MARK not only to understand general technical knowledge but also to act on individualized customer data. This personalized, context-aware support significantly reduces the need for human intervention in routine inquiries, increases operational efficiency, and enhances overall customer satisfaction.

Function calls

Customer: Sim card not working

MARK: To assist you with your SIM card not working, please provide your SIM card's ICCID number (which should be 19-20 digits long). This will help me check its status and determine if there are any issues such as expiration or lack of data.

Customer: *********24107496794

MARK: The SIM card with ICCID *********24107496794 is active and currently has a plan of ***** at a cost of ***. It will not expire until June 27, 2026. If you're experiencing issues, please check the device settings or let me know how I can further assist you!

6 Conclusion

The development and initial testing of MARK demonstrate the potential of RAG-based AI systems to address key challenges in education and technical support. By integrating carefully curated knowledge with LLMs, MARK is able to generate more accurate and trustworthy responses, reducing the risk of hallucination.

Another key benefit of MARK is its ability to support many students without requiring more effort from instructors. The system can handle common student questions, guide them with helpful prompts, and improve over time by learning from user ratings and instructor feedback. These features make MARK an effective tool for both classroom education—such as in engineering courses—and technical support tasks for customers. Its success shows the value of human feedback.

However, there is still room for improvement. MARK depends heavily on the quality and depth of its knowledge base, which means experts must keep updating and reviewing the content. Also, the current system only works with text, limiting its usefulness in subjects like engineering that often rely on images, equations, and diagrams. Future versions of MARK should include the ability to process different kinds of information (multimodal data) and offer more personalized feedback based on each student's needs.

Our testing also shows that the AI system cannot fully replace human support. While MARK is effective in handling common and repetitive queries, many customers still prefer or specifically request interaction with a real person. Some customers can also quickly recognize when they are interacting with an AI chatbot, showing the gap between machine-driven and human support. Such limitations emphasize the need for continued improvement in AI conversational abilities.

References

[1] C. Binkley. College students are still struggling with basic math. professors blame the pandemic. `https://apnews.com/article/college-math-test-help-6cca6a5e873d5aeb5e75b4f94125d48c`, 2023. Associated Press, Accessed: 2025-04-29.

[2] Naep mathematics: National average scores, 2022. National Assessment of Educational Progress (NAEP).

[3] S.D. Sparks. Which nations' students are defying the math anxiety trend? `https://www.edweek.org/teaching-learning/which-nations-students-are-defying-the-math-anxiety-trend/2024/11`, 2024. Education Week.

[4] J.M. Fernández-Luna, J.F. Huete, A. MacFarlane, and E.N. Efthimiadis. Teaching and learning in information retrieval. *Information Retrieval*, 12:201–226, 2009.

[5] N. Ott and D. Meurers. Information retrieval for education: Making search engines language aware. *Themes in Science and Technology Education*, 3:9–30, 2010.

[6] O. Henkel, Z. Levonian, C. Li, and M. Postle. Retrieval-augmented generation to improve math question-answering: Trade-offs between groundedness and human preference. In *Proceedings of the 17th International Conference on Educational Data Mining*, pages 315–320, 2024.

[7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, and S. Agarwal. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[8] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H.W. Chung, C. Sutton, S. Gehrmann, and P. Schuh. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[9] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, and A. Rodriguez. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[10] A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky, A. Low, A. Helyar, A. Madry, A. Beutel, A. Carney, and A. Iftimie. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[11] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, and X. Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[12] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F.L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, and R. Avila. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[13] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y.K. Li, Y. Wu, and D. Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[14] T.H. Trinh, Y. Wu, Q.V. Le, H. He, and T. Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

[15] Y. Chervonyi, T.H. Trinh, M. Olšák, X. Yang, H. Nguyen, M. Menegali, J. Jung, V. Verma, Q.V. Le, and T. Luong. Gold-medalist performance in solving olympiad geometry with alphageometry2. *arXiv preprint arXiv:2502.03544*, 2025.

[16] T. Rasul, S. Nair, D. Kalendra, M. Robin, F. de Oliveira Santini, W.J. Ladeira, M. Sun, I. Day, R.A. Rather, and L. Heathcote. The role of chatgpt in higher education: Benefits, challenges, and future research directions. *Journal of Applied Learning and Teaching*, 6(1):41–56, 2023.

[17] J. Dempere, K. Modugu, A. Hesham, and L.K. Ramasamy. The impact of chatgpt on higher education. *Frontiers in Education*, 8:1206936, September 2023.

[18] G.F.N. Mvondo, B. Niu, and S. Eivazinezhad. Exploring the ethical use of llm chatbots in higher education, 2023. Available at SSRN 4548263.

[19] S. Dakshit. Faculty perspectives on the potential of rag in computer science higher education. In *Proceedings of the 25th Annual Conference on Information Technology Education*, pages 19–24, October 2024.

[20] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y.J. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[21] E. Kasneci, K. Seßler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier, and S. Krusche. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103:102274, 2023.

[22] N. Dziri, E. Kamalloo, S. Milton, O. Zaiane, M. Yu, E.M. Ponti, and S. Reddy. Faithdial: A faithful benchmark for information-seeking dialogue. *Transactions of the Association for Computational Linguistics*, 10:1473–1490, 2022.

[23] L. Zhui, L. Fenghe, W. Xuehu, F. Qining, and R. Wei. Ethical considerations and fundamental principles of large language models in medical education. *Journal of Medical Internet Research*, 26:e60083, 2024.

[24] B. Liu, Z. Cui, S. Hu, X. Li, H. Lin, and Z. Zhang. Llm evaluation based on aerospace manufacturing expertise: Automated generation and multi-model question answering. *arXiv preprint arXiv:2501.17183*, 2025.

[25] C. Zhai, S. Wibowo, and L.D. Li. The effects of over-reliance on ai dialogue systems on students' cognitive abilities: A systematic review. *Smart Learning Environments*, 11(1):28, 2024.

[26] P. Zhang, S. Xiao, Z. Liu, Z. Dou, and J.Y. Nie. Retrieve anything to augment large language models. *arXiv preprint arXiv:2310.07554*, 2023.

[27] K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*, 2021.

[28] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.T. Yih, T. Rocktäschel, and S. Riedel. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[29] P. Béchard and O.M. Ayala. Reducing hallucination in structured outputs via retrieval-augmented generation. *arXiv preprint arXiv:2404.08189*, 2024.

[30] Jason Kirchenbauer and Caleb Barns. Hallucination reduction in large language models with retrieval-augmented generation using wikipedia knowledge, 2024.

[31] C. Niu, Y. Wu, J. Zhu, S. Xu, K. Shum, R. Zhong, J. Song, and T. Zhang. Ragtruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. *arXiv preprint arXiv:2401.00396*, 2023.

[32] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2:1, 2023.

[33] D. Thüs, S. Malone, and R. Brünken. Exploring generative ai in higher education: A rag system to enhance student engagement with scientific literature. *Frontiers in Psychology*, 15:1474892, 2024.

[34] R.M. Perez, M. Shimogawa, Y. Chang, H.A.T. Phan, J.G. Marmorstein, E.S. Yanagawa, and E.J. Petersson. Large language models for education: Chemtask–an open-source paradigm for automated q&a in the graduate classroom. *arXiv preprint arXiv:2502.00016*, 2025.

[35] M. Faghih Shojaei, R. Gulati, B.A. Jasperson, S. Wang, S. Cimolato, D. Cao, W. Neiswanger, and K. Garikipati. Ai-university: An llm-based platform for instructional alignment to scientific classrooms. *arXiv e-prints*, page arXiv:2504, 2025.

[36] I. Ma, A. Krone-Martins, and C. Videira Lopes. Integrating ai tutors in a programming course. In *Proceedings of the 2024 on ACM Virtual Global Computing Education Conference V. 1*, pages 130–136, December 2024.

[37] T. Feng, S. Liu, and D. Ghosal. Courseassist: Pedagogically appropriate ai tutor for computer science education. In *Proceedings of the 2024 on ACM Virtual Global Computing Education Conference V. 2*, pages 310–311, December 2024.

[38] M. Thway, J. Recatala-Gomez, F.S. Lim, K. Hippalgaonkar, and L.W. Ng. Battling botpoop using genai for higher education: A study of a retrieval augmented generation chatbot's impact on learning. *arXiv preprint arXiv:2406.07796*, 2024.

[39] H. Schütze, C.D. Manning, and P. Raghavan. *Introduction to Information Retrieval*, volume 39. Cambridge University Press, Cambridge, 2008.

[40] J. Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*, volume 242, pages 29–48, December 2003.

[41] S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[42] J. Devlin, M.W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, June 2019.

[43] V. Karpukhin, B. Oguz, S. Min, P.S. Lewis, L. Wu, S. Edunov, D. Chen, and W.T. Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, November 2020.

[44] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[45] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2010.

[46] L. Gao, Z. Dai, and J. Callan. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. *arXiv preprint arXiv:2104.07186*, 2021.

[47] K. Sawarkar, A. Mangal, and S.R. Solanki. Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. In *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 155–161. IEEE, August 2024.

[48] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[49] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[50] Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents, 2023.

[51] Michael Günther, Saba Sturua, Mohammad Kalim Akram, Isabelle Mohr, Andrei Ungureanu, Bo Wang, Sedigheh Eslami, Scott Martens, Maximilian Werk, Nan Wang, and Han Xiao. jina-embeddings-v4: Universal embeddings for multimodal multilingual retrieval, 2025.