

# Enhancing Higher Education: Personalized Learning with Large Language Models

## Final Report

Date of Submission: 30.08.2024

### Author:

Sarah Nesti

(CID: 02266658)

(GitHub:edsml-sjn22)

(email: sarah.nesti22@imperial.ac.uk)

### Supervised by:

Dr Marijan Beg

John Francois Van Schalkwyk

A Project Report submitted in fulfilment of requirements for the degree of  
Master of Science in Environmental Data Science and Machine Learning

Department of Earth Science and Engineering

Imperial College London

2024

## Table of Contents

<b>1. Introduction .....</b>	<b>2</b>
<b>1.1 Background and Problem Description.....</b>	<b>2</b>
<b>1.2 Objectives and Research Hypothesis.....</b>	<b>2</b>
<b>1.3 Significance and Originality .....</b>	<b>3</b>
<b>2. Methodology .....</b>	<b>3</b>
<b>2.1 Baseline System Overview .....</b>	<b>3</b>
<b>2.2 Experimental Design.....</b>	<b>4</b>
<b>2.2.1 Question Set .....</b>	<b>4</b>
<b>2.2.2 Evaluation .....</b>	<b>5</b>
<b>2.3 Data Preparation.....</b>	<b>7</b>
<b>2.3.1 Document Loading .....</b>	<b>7</b>
<b>2.3.2 Robustness Testing of System Outputs.....</b>	<b>7</b>
<b>2.3.3 Embedding Function .....</b>	<b>8</b>
<b>2.4 System Optimization .....</b>	<b>9</b>
<b>2.4.1 Retrieval Optimization.....</b>	<b>9</b>
<b>2.4.2 Prompt Optimization and LLM model exploration .....</b>	<b>11</b>
<b>2.4.3 Chunking.....</b>	<b>11</b>
<b>2.5 Task-Oriented Dialogue System .....</b>	<b>13</b>
<b>3. Results .....</b>	<b>13</b>
<b>3.1 Final System Overview.....</b>	<b>13</b>
<b>3.2 Final System Performance and Analysis .....</b>	<b>14</b>
<b>4. Conclusion .....</b>	<b>17</b>
<b>5. Discussion.....</b>	<b>17</b>
<b>Appendix -A .....</b>	<b>19</b>
<b>Appendix -B .....</b>	<b>24</b>
<b>References .....</b>	<b>25</b>

## **Abstract**

This report details the development of an AI-based student support assistant aimed at enhancing educational outcomes in higher education. Addressing the limitations of existing AI solutions, which often provide inaccurate or generic responses, this project implements and optimizes a Retrieval-Augmented Generation (RAG) model. The system is designed to handle various levels of question complexity, optimizing retrieval accuracy for both straightforward information queries and more complex problem-solving tasks. For simpler queries, the model is refined to improve the retrieval of contextually relevant information. For more challenging, multi-step questions, advanced metadata techniques are employed to ensure precise retrieval of model solutions.

Furthermore, the system integrates a task-oriented dialogue component that supports higher-order cognitive tasks by providing interactive feedback and guidance, thereby fostering critical thinking. The effectiveness of this AI assistant is demonstrated through its enhanced ability to deliver accurate, contextually appropriate responses tailored to the specific educational needs of students, ultimately improving their learning experience.

## **1. Introduction**

### **1.1 Background and Problem Description**

Artificial Intelligence (AI) has become an integral part of our daily lives, as demonstrated by tools such as OpenAI's ChatGPT (OpenAI, n.d.). The technology offers aid in writing and brainstorming, and assistance with research and analysis (Ka Yuk Chan & Hu, 2023). Despite its widespread use, AI has not yet been successfully applied to support learning in higher education. Current AI solutions, such as ChatGPT, provide generic answers that are often inaccurate, inadequate or incorrect. This is particularly problematic in university courses, where accuracy and relevance are critical to student understanding and success (Dempere, et al., 2023). Further concerns about the application of AI in higher education are that over-reliance on AI may hinder the development of critical thinking and creativity and also that it misaligns with human values (Ka Yuk Chan & Hu, 2023).

Moreover, the ever-increasing number of students and the dramatic reduction in the time staff can dedicate to individual students have created a pressing need for AI solutions (Barshay, 2020). These challenges necessitate the development of AI systems that can provide precise, context-aware assistance tailored to the specific needs of higher education. The issues of accuracy and transparency of AI-generated information can be addressed by using embedding models that query a vectorised database of relevant data, such as course materials (Choi, 2023). However, a basic implementation of a Retrieval-Augmented Generation (RAG) model that includes such an embedding model developed for a Master's programme at Imperial College has shown that while this approach addresses the issues of relevance and accuracy, it often results in AI answers that are too close to direct solutions. This undermines the learning process by eliminating the need for critical thinking.

### **1.2 Objectives and Research Hypothesis**

The research hypothesis guiding this study is that the implementation of a refined Retrieval-Augmented Generation (RAG) model, combined with a task-oriented dialogue system, will

significantly enhance the relevance and accuracy of AI-generated responses, thereby improving students' understanding and engagement. This enhancement will be reflected in higher-quality retrieval for simpler tasks and will introduce a new level of support for more challenging tasks, such as solving exercises, by offering personalized learning through engaging hints and tailored feedback. The effectiveness of this approach will be assessed using metrics for retrieval quality, relevance, and human evaluations of the system's ability to foster critical engagement and deeper learning.

### **1.3 Significance and Originality**

Existing solutions in the AI education space, provide broad-spectrum responses but lack course-specific customization. Commercial and academic approaches to refining AI responses through embedding models and vectorized databases have shown promise but also highlighted the need for further enhancement to effectively support critical thinking and learning (Ka Yuk Chan & Hu, 2023). This project goes beyond the state-of-the-art by integrating AI in a way that supports the learning process rather than replacing it.

One of the main original contributions of this research is the development of specialized agents within the RAG system that are tailored to different educational tasks. These agents can handle different types of questions differently and can lead task-oriented dialogues, providing detailed explanations, generating hints, and giving feedback. Additionally, the project includes unique enhancements to the RAG setup by exploring and evaluating system configurations to improve the retrieval performance. By focusing on these goals, students are encouraged to engage deeply with the material, ultimately enhancing their learning experience and better preparing them for their future academic and professional pursuits.

## **2. Methodology**

This research develops an AI-based learning support system utilizing a Retrieval-Augmented Generation (RAG) pipeline combined with a task-oriented dialogue system. The approach leverages large language models and course-specific content to provide accurate, context-aware responses while enhancing student interaction and insight. Key processes include data preparation, system configuration, optimisation of the retrieval process and specialized agents, aimed at optimizing the system's ability to handle diverse educational tasks effectively. Additionally, the evaluation metrics, processes, and experimental setup are carefully designed to ensure rigorous testing and validation of the system's performance.

### **2.1 Baseline System Overview**

The chosen approach for developing an effective AI learning support system involves implementing a Retrieval-Augmented Generation (RAG) pipeline, combined with a task-oriented dialogue system. This methodology was inspired by a preliminary RAG system deployed in a Master's course at Imperial College London. It is preferred over alternatives like fine-tuning because it combines the power of large language models with a database of specific course content, ensuring that the information provided is both accurate and relevant. Unlike fine-tuning, which requires extensive retraining and computational resources to adapt a model to specific content, RAG allows for the dynamic retrieval of information from a predefined set of documents, making it more efficient and flexible (Maryamah, et al., 2024).

The system utilizes pre-processed course-specific documents, which have been chunked, embedded into vector representations, and stored in a Chroma vector database (Chroma, 2024). These vectorized chunks, enriched with metadata such as the source document and chunk index, enable the system to efficiently retrieve and process relevant content during query handling. When a student submits a query, it is embedded using the same model that processed the documents. The embedded query is then matched against the vectorized database using a retriever that applies a similarity search strategy to find the most relevant documents. Key parameters in this retrieval process, such as the method used to search (`search_type`) and the number of documents retrieved (`k`), are configured to strike a balance between retrieval accuracy and efficiency. After retrieving the relevant documents, the system generates a prompt that incorporates the retrieved information. This prompt is then sent to an LLM API, which generates responses based on the most relevant content available. The baseline architecture, illustrated in the flowchart in Figure 1, outlines the core components and their interactions, providing a foundation for the system's ability to deliver accurate, context-aware responses.

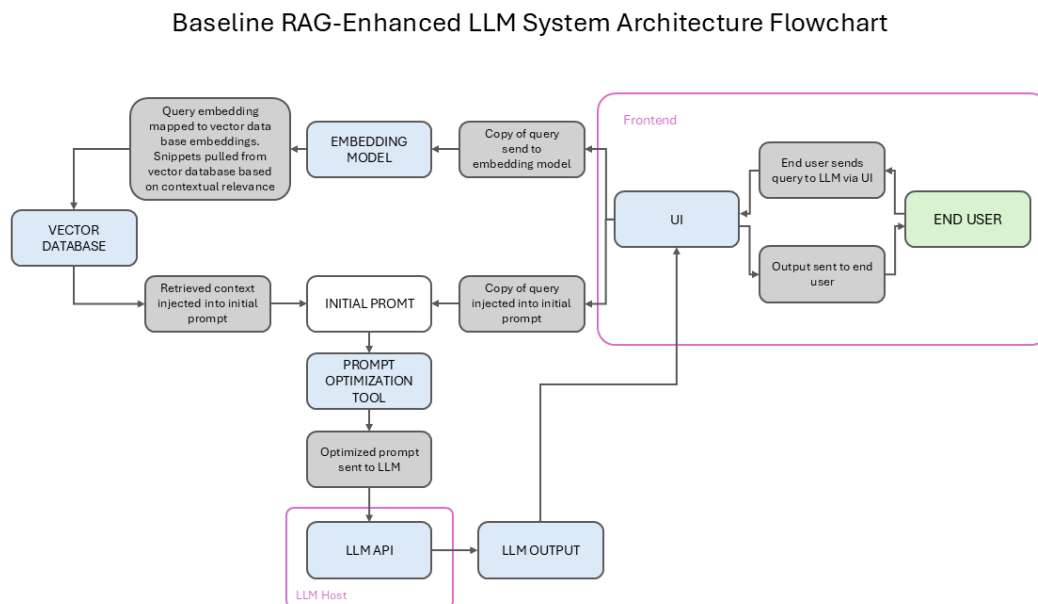


Figure 1: Flowchart of the baseline Retrieval-Augmented Generation (RAG) system architecture. The user query is processed through an embedding model and a prompt optimization tool, which retrieves relevant context from a vector database. The optimized prompt is then sent to the LLM, with the final response delivered back to the user via the UI. (Choi, 2023)

## 2.2 Experimental Design

### 2.2.1 Question Set

The experimental setup is designed to rigorously test and validate the system's performance under realistic conditions. The experiment addresses two distinct types of educational queries—Type A and Type B—which align with different difficulty levels of Bloom's Taxonomy, a framework widely used in educational settings to categorize the cognitive complexity of learning objectives and assessment (Jones, et al., 2009).

**Type A questions** are contextual and understanding-based queries that correspond to the Lower Order Cognitive Questions (LOCQ) in Bloom's Taxonomy, specifically at the knowledge and

comprehension levels. In this category, students are expected to recall and understand facts without necessarily applying or analysing them in new contexts. The system must provide concise, factual responses that reinforce the student's understanding of lecture material.

**Type B questions** arise during problem-solving activities and correspond to the Intermediate Order Cognitive Questions (IOCQ) and Higher Order Cognitive Questions (HOCQ) in Bloom's Taxonomy, covering application, analysis, synthesis, and evaluation levels. Type B questions require students to apply their knowledge to new situations, analyse problems, and create or evaluate solutions.

To systematically assess the system's capabilities, a set of 10 questions was developed for each question type (Type A and Type B). These questions, stored in JSON files, were carefully crafted to cover a range of difficulty levels and educational tasks, ensuring a comprehensive evaluation of the system's functionality (see Appendix A).

### 2.2.2 Evaluation

To efficiently and objectively evaluate the system's performance, the goal is to introduce automated evaluation methods that can reliably mimic human judgment. Given the challenges associated with manually grading all responses, particularly as the system scales, it is crucial to establish evaluation metrics that closely correlate with human assessments.

The system needs to handle two distinct types of educational queries, each with different evaluation needs:

**Type A questions:** the evaluation metric must effectively assess the semantic relevance of the system's responses, ensuring that the answers are contextually appropriate and contribute to enhancing the student's understanding of the material.

**Type B questions:** the evaluation metric needs to precisely measure the accuracy of the system's responses in comparison to model solutions, which is crucial for effectively guiding students through problem-solving tasks.

#### Evaluation Process:

To validate the hypothesis, the system's responses to a set of 10 questions per type were compared against human-graded answers using four different metrics:

1. **Sequence Matching:** Assesses the exact alignment between the generated and expected answers, focusing on precision.
2. **LLM-Based Similarity and Relevance:** Utilizes a large language model to score the relevance of the generated response compared to the expected solution.
3. **Embedding Similarity:** Measures the cosine similarity between the generated and reference answers, capturing semantic meaning and relevance.
4. **Augmentation Accuracy:** Calculates the token-level overlap between the generated answer and the reference text, indicating content overlap.

The results, illustrated in the heatmap (Figure 2), show that Embedding Similarity most closely aligned with human evaluation for Type A questions. For Type B questions, Sequence Matching was most consistent with human judgment.

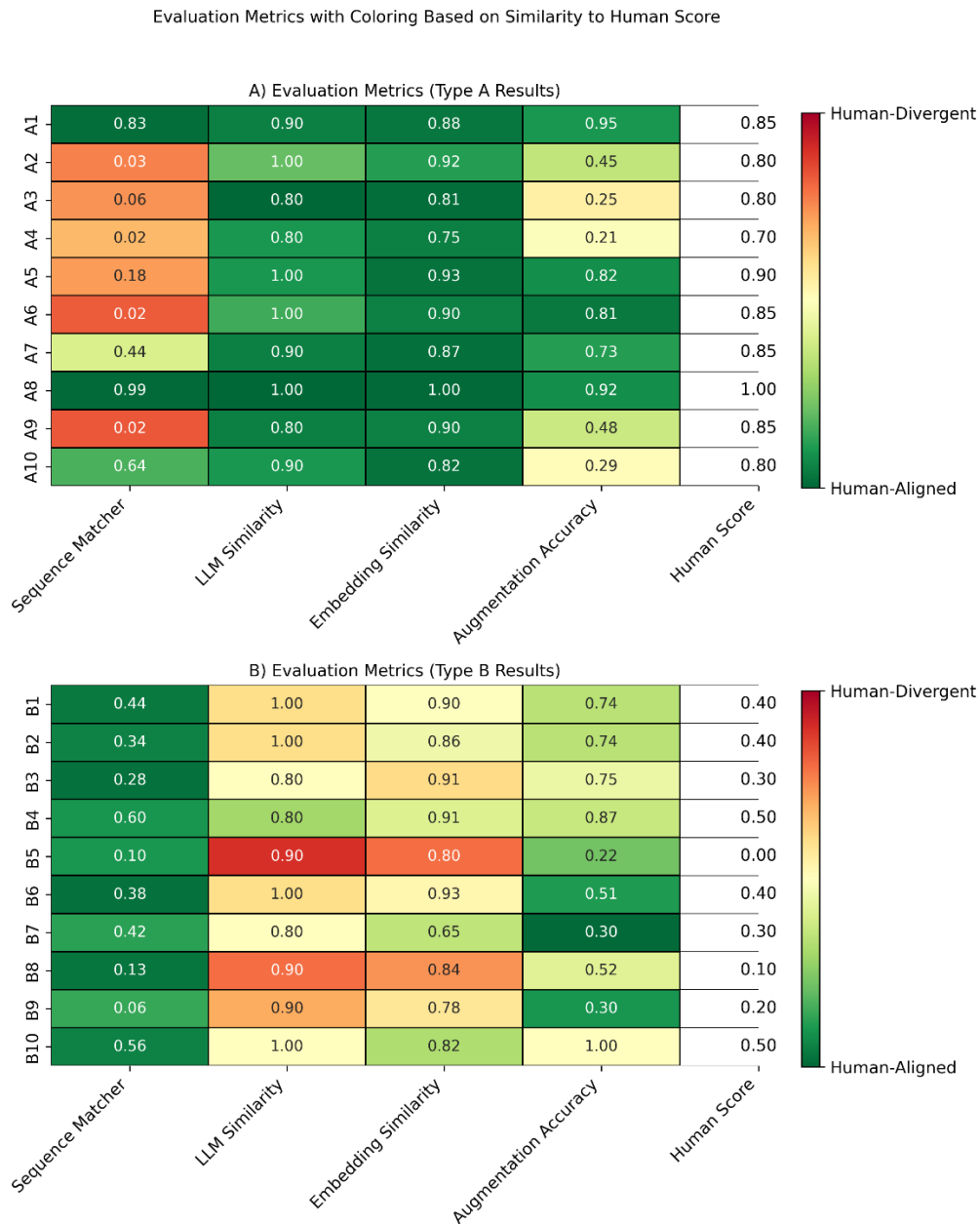


Figure 2: The heatmaps show evaluation results of different evaluation metrics for 10 questions each of Type A and B. In subplot A (Type A), Embedding Similarity aligns most closely with human scores, while in Subplot B (Type B), Sequence Matcher shows the highest alignment.

These metrics will be used in subsequent evaluations to measure the impact of system improvements and optimizations.

Unlike straightforward retrieval tasks, the effectiveness of a task-oriented dialogue system in an educational setting is deeply rooted in its ability to interact meaningfully with students, guiding them through complex problem-solving processes. A full assessment of such a system typically requires real-life student surveys and interaction studies, which lie beyond the scope of this

thesis. Therefore the evaluation of the dialogue system in this study is supported by two key approaches: alignment with established theoretical frameworks and human evaluation.

The design of the system is firmly grounded in VanLehn's (2006) model of tutoring systems, which are characterized by two interconnected loops: an outer loop responsible for selecting appropriate tasks and an inner loop that provides step-by-step feedback and hints as the student progresses through a task. By emulating this structure, the system developed in this study offers a robust mechanism for educational support.

It is further evaluated through direct human interaction via the user interface (UI). In this approach, a variety of student interactions are evaluated by posing questions of varying difficulty, similar to those used in the RAG evaluation. The system's responses are then assessed for their adherence to the tutoring model and their pedagogical appropriateness. This involves examining how well the system provides feedback, offers hints, and adjusts its guidance based on the student's input.

## **2.3 Data Preparation**

### **2.3.1 Document Loading**

The data preparation phase is critical for ensuring that the Retrieval-Augmented Generation (RAG) system can efficiently retrieve and generate accurate responses. This process involves three main steps: loading course documents, chunking the content into manageable segments, and embedding these chunks into a vectorized database for retrieval. The first step in data preparation involves loading the course materials, which include lecture notes and exercise notebooks, initially stored in various formats, particularly Jupyter Notebooks (Jupyter, 2024). In this project 19 notebooks (2,54MB total) of lectures and exercises with solutions were embedded. The content from these notebooks, including both markdown and code cells, is extracted and consolidated to ensure that all relevant educational material is available for the system's retrieval process.

### **2.3.2 Robustness Testing of System Outputs**

To ensure the reliability of the system's performance evaluations, robustness testing was conducted over 10 runs of the baseline model. Initially, the system's outputs exhibited significant variability across these runs, as illustrated in the subplot a in Figure 3. This inconsistency raised concerns about the influence of randomness on the evaluation results. To address this, the temperature setting of the LLM was adjusted to 0, reducing the randomness in the model's responses. Post-adjustment, the system's outputs became significantly more consistent, as shown in the subplot b in Figure 3. This improvement in robustness allows us to confidently assume that observed enhancements in system performance are genuine and not due to outliers. Moving forward, the temperature will be kept at 0 for all evaluations, with 10 runs conducted as a precautionary measure, ensuring that the results are reliable.



Box Plots of Similarity Scores Across 30 Runs for 10 Type B Questions Using LLaMA 3 (70B)

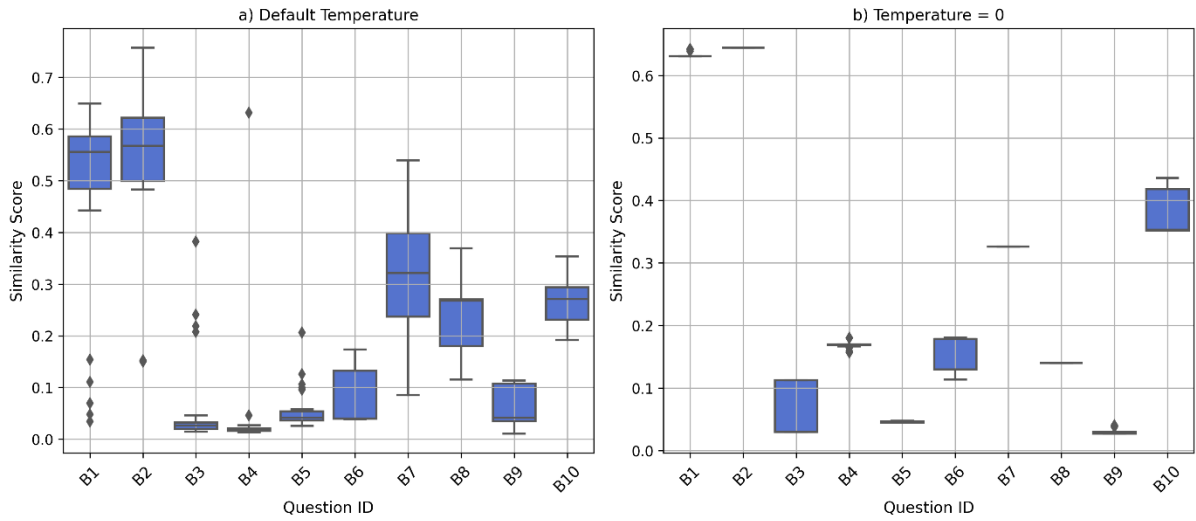


Figure 3: Box plots of similarity scores across 30 runs for 10 Type B questions using the llama 3 (llama3:70b-instruct-q8\_0) model. (a) Results with the default temperature setting, demonstrating higher variability and less consistency. (b) Results with a temperature setting of 0, showing reduced variability and enhanced accuracy.

### 2.3.3 Embedding Function

The documents are first chunked and then embedded into vectorized representations using a selected embedding model. The embedded vectors are stored in a Chroma vector database (Chroma, 2024), optimized for efficient querying. Metadata, such as the source document and chunk index, is associated with each chunk to enhance both storage and retrieval processes, ensuring that relevant content can be quickly and accurately identified during the query process. The effectiveness of the embedding model is crucial for the accuracy of the retrieval process. It was hypothesized that different embedding models would yield varying levels of performance in similarity scoring.

To test this hypothesis, four different models were evaluated using the framework SentenceTransformerEmbeddings (Hugging Face, 2024), in addition to the baseline model from langchain\_community.embeddings.fastembed (Langchain, n.d.) (BAAI/bge-base-en-v1.5, n.d.). Due to limitations in testing multiple embedding models with FastEmbedEmbeddings, the switch was made to the SentenceTransformerEmbeddings framework for broader evaluation.

The four models tested with SentenceTransformerEmbeddings were:

1. **all-MiniLM-L6-v2**: used for tasks like clustering or semantic search (all-MiniLM-L6-v2, n.d.)
2. **msmarco-distilbert-base-tas-b**: A BERT-based model optimized for the task of semantic search. (msmarco-distilbert-base-tas-b, n.d.)
3. **stsb-roberta-base-v2**: used for tasks like clustering or semantic search. (stsb-roberta-base-v2, n.d.)

4. **BAAI/bge-base-en-v1.5**: Tested within the SentenceTransformerEmbeddings framework but also used as the embedding model in the baseline FastEmbed framework. (BAAI/bge-base-en-v1.5, n.d.)

Each model was tested by running both the Type A and Type B question sets 10 times. For each question, the mean similarity score was calculated across the 10 runs. The overall performance of each model was then visualized using box plots (see Figure 5), showing the distribution of similarity scores across the 10 questions for each question type.

Performance Comparison of Embedding Models on RAG Across Two Question Types and 10 runs each

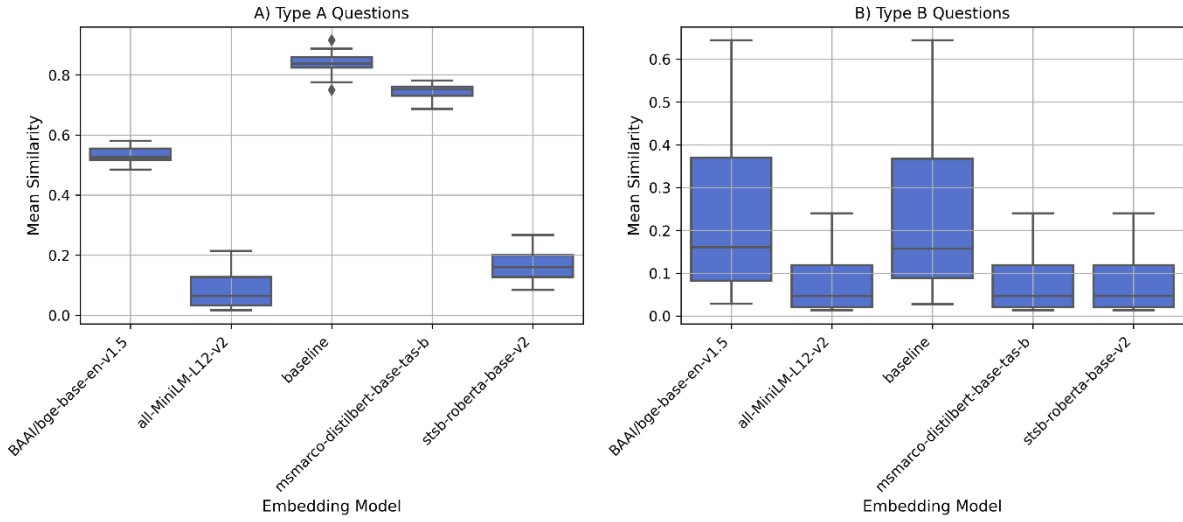


Figure 4: Box plots of mean similarity scores across 10 questions of Type A (Subplot a) and Type B (Subplot b) for different embedding models. The overall figure demonstrates the impact of different embedding models on RAG performance, highlighting the consistency and reliability of each model in relation to expected answers.

The results, presented in Figure 4, indicate that while several models performed competently, the FastEmbedEmbeddings (BAAI/bge-base-en-v1.5) model consistently outperformed the others in both Type A and Type B questions. This difference is particularly pronounced for Type A questions, which require strong contextual understanding. This outcome aligns with expectations, as Type A questions are pure retrieval tasks, evaluated using the same embedding model that generated the database embeddings. Thus, a superior embedding model directly leads to better retrieval results.

Given these findings, the FastEmbedEmbeddings (BAAI/bge-base-en-v1.5) model was chosen for further development and testing. Its superior retrieval accuracy since the retrieval accuracy is critical for the overall effectiveness of the AI learning support system.

## 2.4 System Optimization

### 2.4.1 Retrieval Optimization

#### Retriever k

The retrieval parameter k was explored to determine its effect on the RAG system's performance. Given the emphasis on contextual understanding in Type A queries, different k values were tested

to find the optimal setting that balances the retrieval of relevant information without overwhelming the system with too much data. For Type B questions, the retrieval parameter  $k$  was set to 1 based on the requirement for precise problem-solving guidance, where retrieving the single most relevant solution is critical. As a result, the focus of the evaluation for Type A questions was on exploring a range of  $k$  values to identify the most effective setting, which is illustrated in Figure 5.

Impact of Different  $k$  Values on Retriever Performance Across 10 Questions of Type A

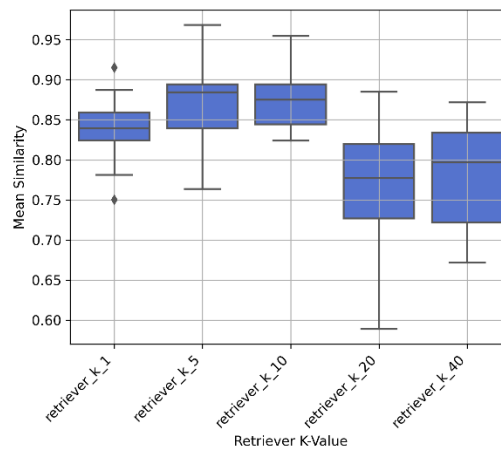


Figure 5: Box plots of mean similarity scores across 10 Type A questions for different retriever  $k$  values. Each box plot represents the mean similarity scores calculated over 10 runs per question, highlighting the impact of varying  $k$  on retrieval performance.

The evaluation results indicate that a  $k$  value of 10, despite not exhibiting the highest mean, demonstrates the highest minimum value, with the lowest quartile consistently above 0.825. Therefore even the least effective responses maintains a strong baseline performance, indicating robustness across varying difficulty levels.

## Search Type

ChromaDB was selected as the primary vector database due to its robust performance, ease of integration with large language models (LLMs), and native support within the LangChain (Langchain, n.d.) framework. However, its lack of hybrid search capabilities—combining vector and keyword-based retrieval—presented a challenge. Initial attempts to implement keyword search and develop a custom hybrid search mechanism within ChromaDB proved too complex and time-consuming for the scope of project.

Consequently, the project focused on evaluating ChromaDB's built-in search functionalities, specifically Maximum Marginal Relevance (MMR), similarity search, and similarity threshold search. In this Evaluation both similarity search and similarity threshold search performed similarly, with mean similarity scores close to 0.9, while MMR showed slightly lower performance. Given the negligible difference, the simpler similarity search method was chosen for further development to ensure an efficient retrieval process. Detailed results and the corresponding box plot are available in Appendix A.

### 2.4.2 Prompt Optimization and LLM model exploration

The exploration of Language Learning Models (LLMs) and prompt versions was a critical component in optimizing the performance of the Retrieval-Augmented Generation (RAG) system. Figure 6 presents a comprehensive comparison of various LLMs—GPT-3.5 Turbo, GPT-4, GPT-4o mini, and Llama3:70b-instruct-q8\_0—across different prompt versions. The figure highlights how the mean similarity scores, which serve as a measure of retrieval performance, vary depending on the combination of LLM and prompt version used.

Contrary to expectations, the combination of the baseline prompt and the Llama 3 model outperformed more advanced models and prompts in terms of mean similarity. This finding challenges the assumption that increased model and prompt complexity necessarily translates to better retrieval performance, indicating that simpler configurations may offer more robust results in certain contexts.

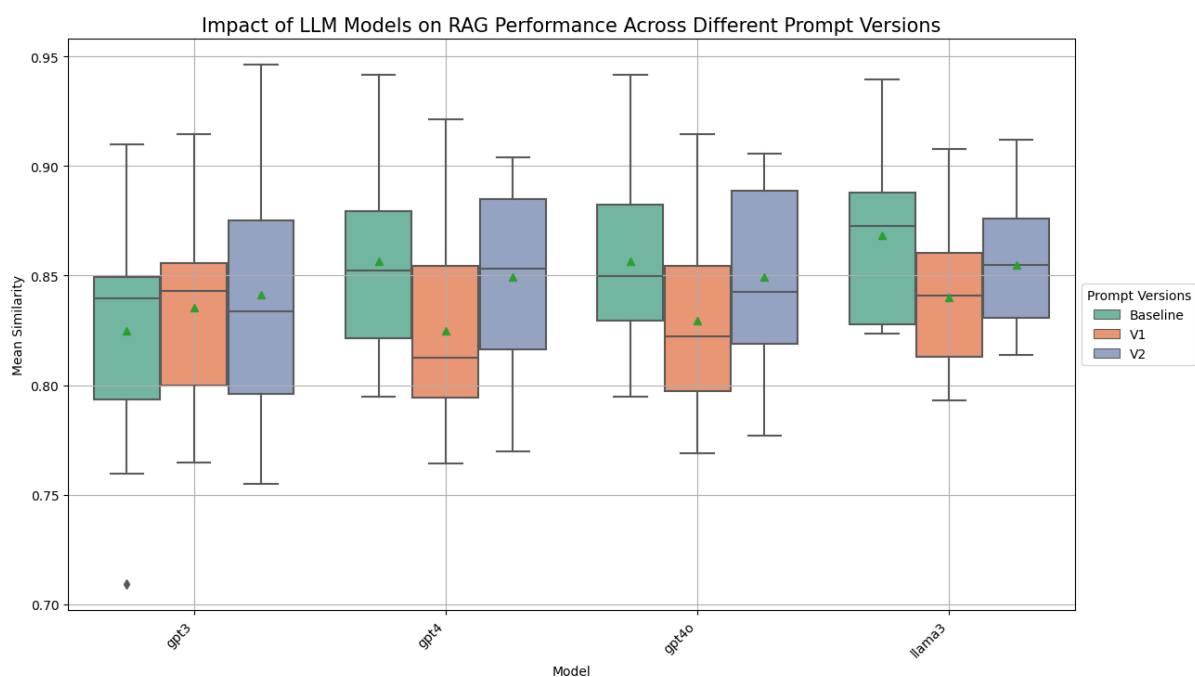


Figure 6: Box plots showing the impact of different prompt versions and various language models on the RAGs retrieval performance as measured by mean similarity scores. The LLMs evaluated include GPT-3.5 Turbo, GPT-4, GPT-4o mini, and llama3:70b-instruct-q8\_0. The plot highlights that the baseline plot and the llama 3 model are the best performing combination.

### 2.4.3 Chunking

Efficient document chunking plays a crucial role in optimizing retrieval performance within the system. The chunking strategy not only influences the relevance of retrieved content but also significantly impacts the overall accuracy of the system. The primary objective of this section is to explore various chunking strategies and identify the most effective approach for enhancing the retriever's performance, particularly across different types of questions.

It was hypothesized that chunking strategies incorporating structural elements of the documents, such as headings and subheadings, would outperform simpler strategies that rely solely on uniform chunk sizes. Conversely, for Type B questions, which are more detailed and often involve exercises, it was predicted that chunking strategies preserving the logical structure

of the content (e.g., by keeping entire exercises intact) and leveraging metadata to separate questions from solutions would lead to enhanced retrieval accuracy.

Several chunking strategies were implemented and tested:

1. **Baseline Chunking:** Documents are divided by lines that start with # (for headings) and by a max chunk size of 800
2. **Subheading-Based Chunking:** Documents are chunked by subheadings ('#', '##', '###') to preserve the document's structure
3. **Enhanced Chunking for Type B:** Combined cell and subheading-based chunking, using metadata to differentiate between questions and solutions, aimed specifically at improving Type B retrieval accuracy
4. **Small Chunk Size (300 tokens):** fine granularity chunking by size
5. **Medium Chunk Size (800 tokens):** moderate chunk size
6. **Large Chunk Size (1300 tokens):** larger chunks size
7. **Optimal Chunk Size with Overlap:** the most effective chunk size identified in previous experiments with an 80-token overlap to maintain context at chunk boundaries

The outcomes of these experiments are depicted in Figure 8, which presents two subplots illustrating the impact of different chunking strategies on retriever performance across Type A and Type B questions.

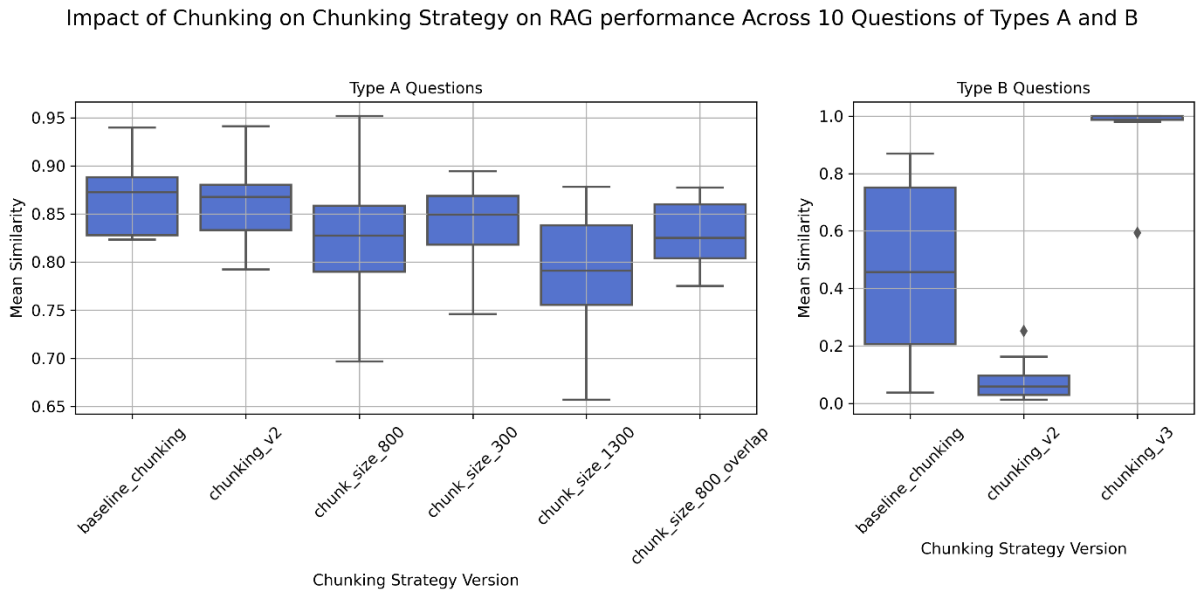


Figure 7: Impact of different chunking strategies on retriever performance across 10 questions of Types A and B. The left panel displays results for Type A questions, while the right panel focuses on Type B questions.

**Type A Questions:** Subplot (a) in Figure 8 demonstrates that, contrary to expectations, the subheading-based chunking strategy (chunking\_v2) did not outperform the baseline chunking strategy. While it was anticipated that leveraging document structure would enhance retrieval accuracy, this approach yielded results comparable to those of simpler chunking methods. Therefore for Type A questions the baseline model is chosen for the further course of this project.

**Type B Questions:** Subplot (b) in Figure 8 highlights the significant influence of chunking strategies on Type B questions. The enhanced chunking method (chunking\_v3), which

maintained the integrity of entire exercises and utilized metadata to differentiate between questions and solutions, achieved the highest mean similarity scores and will therefore be used for Type B retrieval in the final model.

## **2.5 Task-Oriented Dialogue System**

The task-oriented dialogue system is designed to provide targeted educational support within a Higher Education course by leveraging large language models (LLMs). In a first step the system classifies student queries into two categories: contextual understanding (Type A) and problem-solving (Type B). This classification enables precise, context-aware responses that align with the student's current learning needs, facilitating more effective educational interactions. The system's design integrates Bloom's Taxonomy, supporting cognitive processes from basic recall to complex problem-solving (Jones, et al., 2009). For Type A queries, the system retrieves relevant information and provides detailed explanations, reinforcing the cognitive processes of remembering and understanding as per Bloom's Taxonomy. For Type B queries, the system guides students through problem-solving steps, offering hints and feedback that encourage higher-order thinking such as application, analysis, and synthesis. The system's effectiveness in promoting cognitive engagement at multiple levels underscores its value as an educational tool.

The system is implemented using React (React, 2024), which provides a seamless and interactive user interface, allowing students to easily engage with the system. The functionality was primarily evaluated through targeted test questions, focusing on the accuracy and relevance of hints and feedback provided by the system. The system was assessed for its alignment with VanLehn's (2006) model of tutoring systems, which includes an outer loop for task selection and an inner loop for providing step-by-step feedback and hints.

## **3. Results**

### **3.1 Final System Overview**

The final system integrates a RAG architecture, optimized specifically for retrieval, and enhanced with a LLM to support an adaptive learning environment. This system efficiently handles and responds to student queries by leveraging a combination of natural language processing techniques, state management, specialized databases and retrieval techniques tailored to different types of queries.

The process begins with the end user submitting a query through the user interface (UI). The system then classifies the query into one of two types: Type A, which typically requires retrieval of contextually relevant information, or Type B, which involves step-by-step problem-solving guidance. For each question type, a separate vector database has been created, optimized for the specific retrieval needs of that type. Type A queries access a database that is finely tuned for context-rich information retrieval, utilizing a k=10 retriever to ensure accurate and contextually appropriate responses. These responses are further refined through prompt optimization before being delivered to the user. For Type B queries, the system adopts a more interactive approach, utilizing a separate vector database designed to retrieve the model solutions. If a model solution has already been identified, the system checks whether it is awaiting further input from the student. If input is awaited, the student's response is evaluated, and the system provides feedback or hints as necessary, guiding the student through the problem-solving process step by

step. This approach ensures that the student receives targeted assistance, maintaining engagement and promoting learning. For this the system includes comprehensive state management to track the progress of interactions, including the current step in Type B queries, whether input is awaited, and the status of student responses. This state management allows the system to dynamically adjust its responses based on the student's progress, ensuring a seamless and personalized learning experience.

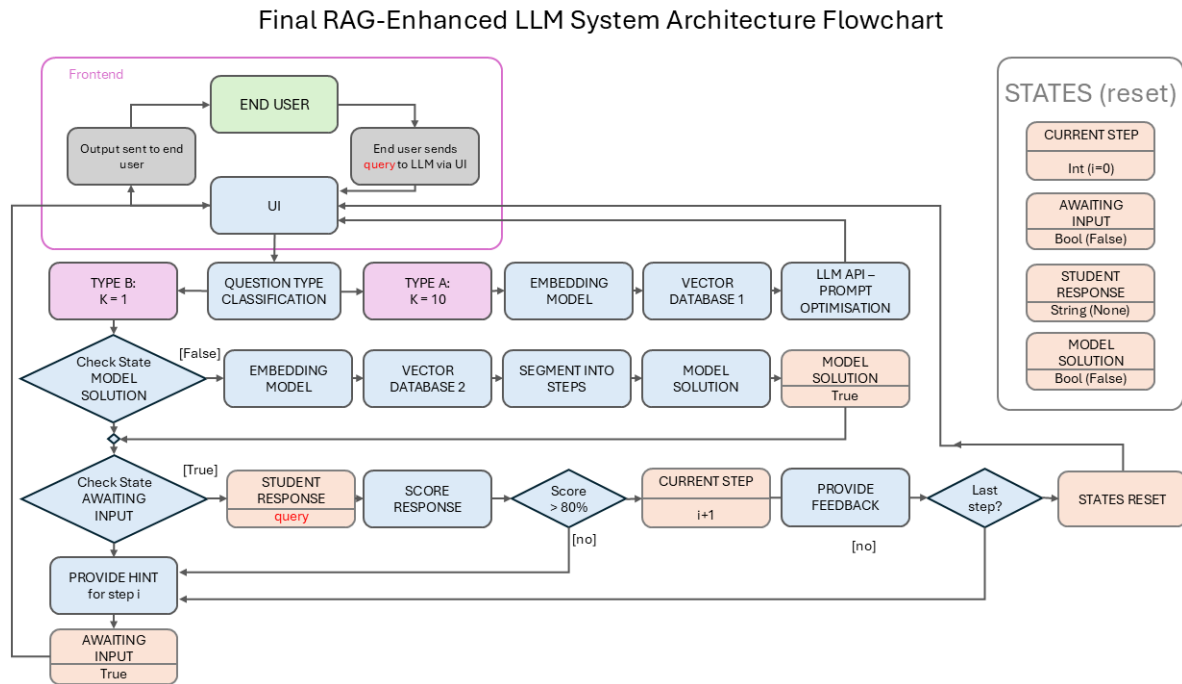


Figure 8: Flowchart of the final RAG-enhanced LLM system architecture. The diagram illustrates the process flow for handling user queries, including the classification of queries into Type A or Type B, retrieval operations using embedding models and vector databases, and state management for providing iterative feedback and guidance.

### 3.2 Final System Performance and Analysis

The final model is developed by integrating and refining the various strategies explored in the methodology. This model is then compared to the baseline system to evaluate improvements in performance.

Figure 9 presents a comparison of the mean similarity scores between the baseline and final systems across Type A and Type B questions. In subplot (a), the results for Type A questions show that the final system achieves a higher median similarity score than the baseline model, with reduced variability, indicating more consistent performance. This improvement, while modest, reflects the effectiveness of the retrieval parameter adjustments and the temperature setting in refining the system's ability to retrieve contextually relevant information.

In subplot (b), the results for Type B questions highlight a substantial improvement in performance with the final system. The near-perfect mean similarity scores, with only one outlier where the question and model solution were retrieved together, demonstrate the system's ability to accurately retrieve model solutions, which is crucial for the task oriented system to function correctly.

Performance Comparison of Baseline vs Final System for Two Question Types and 10 runs each

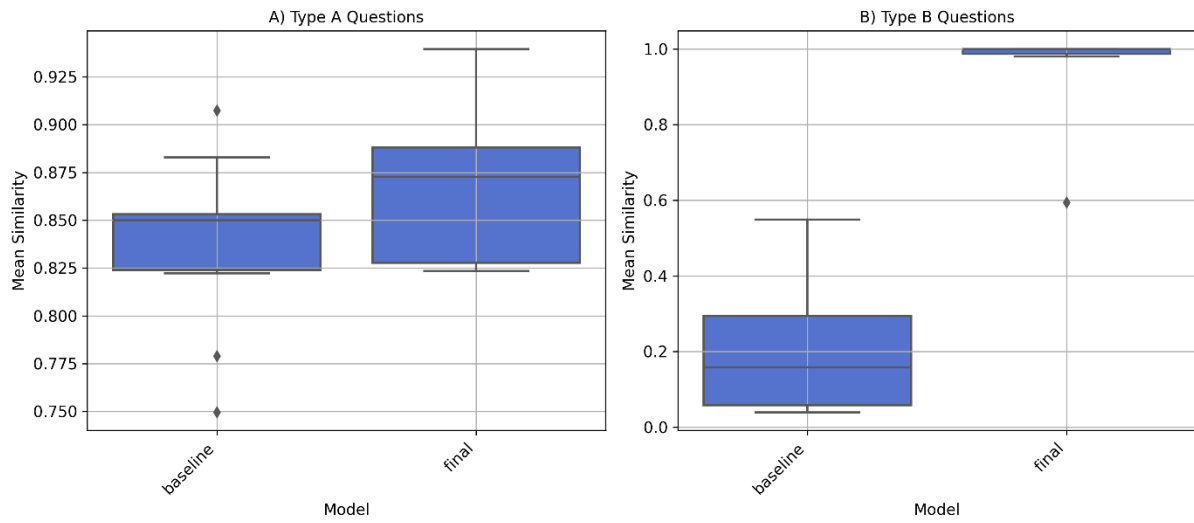


Figure 9: The boxplots depict the mean similarity scores for Type A and Type B questions across 10 runs. For Type A questions, the final system demonstrates an improvement in mean similarity compared to the baseline system, as evidenced by the higher median and reduced variability. For Type B questions, the final system achieves near-perfect mean similarity, with minimal variability, while the baseline system shows lower performance with greater variance.

Furthermore, Figure 10 provides a screenshot of the user interface (UI), which illustrates the final system's capability to manage both Type A and Type B queries. The example presented is specifically crafted for showcasing and evaluating the system's performance in responding to Type A queries and in guiding a student through solving an exercise. The system breaks down the model solution into sub steps and delivers hints and feedback as needed. This example demonstrates the system's ability to not only retrieve information but also to support students



interactively, guiding them through the problem-solving process in a structured and educationally beneficial manner.

UniGPT

U

What are the important characteristics of Algorithms?

The important characteristics of algorithms are:

1. **Precise and unambiguous**: Clear in all aspects, with every step meaning only one thing.
2. **Well-defined inputs**: Expected inputs (zero or more) must be clearly explained.
3. **Well-defined outputs**: The output (one or more) of the algorithm must be well defined.
4. **Finite**: The algorithm must terminate after a finite time.
5. **Feasible**: Implementation must be possible with the available resources.
6. **Language independent**: Ideally, it should not contain language-specific features.
7. **Deterministic**: It gives the same output for the same inputs.

help me write a function bin2dec(x2) that converts a binary string (e.g., '1011.101') to its decimal equivalent as a float? The input might not always include a fractional part.

Hint

Think about how you would separate the whole number part from the decimal part of a number when it is given as a string. How can you identify where the whole number part ends and the decimal part begins?

Please provide your suggested solution for this step.

def bin2dec(x2):  
if '.' in x2:  
int=until  
else:  
int, frac = x2, '0'

Score: 60%

You're close, but there are a few issues with your code.

In the 'if' branch, you have 'int=until', which doesn't make sense in this context. You probably meant to assign the part before the decimal point to a variable, like 'integer2'.

In the 'else' branch, you have 'int, frac = x2, '0'', but this is not correctly assigning the integer and fractional parts of the binary number.

Take another look at the correct solution and see if you can fix these issues. You're on the right track!

Rating: 60%

def bin2dec(x2):  
if '.' in x2:  
integer2, fractional2 = x2.split('.')  
else:  
integer2, fractional2 = x2, '0'

Score: 95%

Your code is almost identical to the correct answer, which is great! The only difference I can see is that you have an extra indentation in your code, but it doesn't affect the functionality.

Rating: 95%

Model Solution

```

**Step 1: Handle input string**
...
if '.' in x2:
    integer2, fractional2 = x2.split('.')
else:
    integer2, fractional2 = x2, '0'
...

```

Hint

Think about how you would convert a binary number to decimal manually. You would take each digit in the binary number, multiply it by a power of 2 based on its position, and then add up all those values. How can you translate this process into code? What operation would you use to raise 2 to a certain power, and how would you iterate over the digits in the binary number?

Type A Question  
Level: Understanding

Type B Question  
Level: Application

Student engagement through hints

Personalized feedback

Once Student reached score threshold, the model solution for that step is revealed and the student is given a hint for the next step

Figure 10: Demonstration of the task oriented dialogue system answering Type A with a simple retrieved answer from the database and Type B questions by engaging the student through smaller steps, hints and feedback.

## 4. Conclusion

This research has significantly advanced the capabilities of a Retrieval-Augmented Generation (RAG) system, demonstrating its potential to enhance adaptive learning environments.

**Conclusion 1: Retrieval Optimisation** focuses on the RAG system's overall improvements, particularly in handling Type A and Type B queries. For Type A questions, modest gains are observed by optimizing the retrieval parameter  $k$  to 10 and setting the LLM temperature to 0, which together improve retrieval accuracy by ensuring that the most relevant documents are prioritized. The enhancements made for Type B queries, which involve more complex problem-solving tasks, show substantial performance improvements. By removing LLM prompt optimization and refining the retrieval process (setting  $k$  to 1), the system effectively retrieves the most relevant information. The embedding of model solutions within the metadata of corresponding exercise chunks ensures near perfect retrieval results, which is crucial for supporting problem solving assistance.

**Conclusion 2: Educational Impact Enhancement** integrates a task-oriented dialogue system into the RAG architecture. Supported by educational literature such as Bloom's Taxonomy (Jones, et al., 2009), and drawing from insights into effective tutoring systems (VanLehn, 2006), it is evident that mere retrieval of information is insufficient for addressing higher-order cognitive tasks. By implementing a dual-loop feedback mechanism, the system engages students in interactive dialogue, providing feedback and strategic hints that align with best practices in tutoring systems. This design fosters deeper learning, encouraging students to move beyond simple recall to more complex cognitive activities, such as analysis and evaluation.

This dual focus on improving retrieval accuracy and enhancing educational impact highlights the potential of this approach to transform adaptive learning environments, offering both immediate technical benefits and long-term educational value.

## 5. Discussion

This research marks a significant step forward in the development of adaptive learning systems, particularly through the integration of a Retrieval-Augmented Generation (RAG) architecture enhanced by a task-oriented dialogue system. However, despite the advancements achieved, several limitations and areas for future exploration remain.

The performance improvements observed for Type A questions, while notable, suggest that there is still considerable room for enhancement. One limitation is the reliance on traditional retrieval methods, which, despite optimizations, have not fully addressed the complexities of more nuanced queries. To overcome this, future research could explore the implementation of hybrid search methods, which combine the strengths of both keyword-based and semantic search techniques. Hybrid search has the potential to significantly improve retrieval accuracy by better understanding the intent behind queries and by leveraging a broader range of data sources. Additionally, further refining the chunking strategy could contribute to these improvements. While making more sophisticated use of metadata and implementing an improved structural chunking strategy or incorporating more detailed contextual information into the metadata was successful for Type B questions, further refinement is necessary for Type A questions. Another remaining question is whether the observed success of the baseline prompt with the Llama 3

model reflects its actual superiority or stems from the limitations inherent in the evaluation metric used.

The integration of a task-oriented dialogue system into the RAG framework has the potential to greatly enhance the educational experience by moving beyond simple information retrieval to actively engaging students in the learning process. Educational literature strongly supports the use of dialogue, feedback, and hints as tools for promoting critical thinking and deeper understanding (VanLehn, 2006). By embedding these elements into the system, this research has laid the groundwork for a more interactive and responsive learning environment that can adapt to the needs of individual students. However, one of the key limitations is the lack of long-term evaluation within a real-world classroom environment. While the system has been designed with educational theory in mind, particularly drawing on principles such as Bloom's Taxonomy, its actual impact on student learning remains to be fully assessed. Empirical studies need to be conducted to measure the educational impact of the dialogue system, focusing specifically on how it affects student engagement, learning outcomes, and the development of critical thinking skills over time. For this the systems speed would need to be optimised. Future work could also focus on refining the dialogue system to handle a wider variety of question types and learning scenarios, as well as integrating more advanced natural language processing techniques to improve the quality of interactions. Moreover, expanding the system's capabilities to include sources and adaptive learning paths that respond to student performance could further enhance its educational value, providing personalized learning experiences that are both effective and engaging.

## Appendix -A

Questions of Type A (Different levels of complexity where chosen)

ID	Question	Expected Answer
A1	What are the important characteristics of Algorithms?	<ul style="list-style-type: none"> <li>• <b>Precise and unambiguous:</b> Clear in all aspects. Every step must mean only one thing.</li> <li>• <b>Well-defined inputs:</b> Expected inputs (zero or more) must be clearly explained.</li> <li>• <b>Well-defined outputs:</b> The output (one or more) of the algorithm must be well defined.</li> <li>• <b>Finite:</b> The algorithm must terminate after a finite time.</li> <li>• <b>Feasible:</b> Implementation must be possible with the available resources.</li> <li>• <b>Language independent:</b> Ideally, it should not contain language-specific features.</li> <li>• <b>Deterministic:</b> It gives the same output for the same inputs.</li> </ul>
A2	What is the <i>tilde approximation</i> of a particular algorithm?	<ul style="list-style-type: none"> <li>• In asymptotic analysis of algorithms, we attempt to predict the execution time as a function of input size <math>N</math> for large values of <math>N</math> (<math>N \rightarrow \infty</math>).</li> <li>• Let us assume that the function that maps the input size <math>N</math> to the execution time is <math>f(N)</math>.</li> <li>• We define tilde approximation <math>\sim g(N)</math> by simply dropping the lower order terms, and write <math>f(N) \sim g(N)</math>.</li> <li>• Mathematically, we can express it as:  <math display="block">\lim_{N \rightarrow \infty} \frac{f(N)}{g(N)} = 1</math> <math display="block">\lim_{N \rightarrow \infty} \frac{g(N)}{f(N)} = 1</math> </li> <li>• Example: <math>f(N) = N^3 + N^2 + N</math>, <math>g(N) = N^3</math></li> <li>• Tilde approximation: <math>f(N) \sim N^3</math></li> </ul>
A3	Give me examples of algorithms for each of the different big-O.	<p><b>Constant: <math>O(1)</math></b>  Usually a simple statement: <code>a[i]</code>  <code>result = operand_1 + operand_2</code></p> <p><b>Logarithmic: <math>O(\log(N))</math></b>  An example is binary search - we search for an element <math>x</math> in an already sorted sequence:  Compare <math>x</math> with the middle element of the sequence.  If the middle element is the same as <math>x</math>, return the middle index.  Else, if <math>x</math> is greater than the middle element, apply the algorithm to the right half.  Else, if <math>x</math> is less than the middle element, apply the algorithm to the left half.</p> <p><b>Linear: <math>O(N)</math></b>  Iterate over a sequence element-by-element using a single loop.  Examples: search, minimum, maximum, etc.</p> <p><b>Quadratic: <math>O(N^2)</math></b>  Mostly includes examples with double loops.</p> <p><b>Cubic: <math>O(N^3)</math></b>  Mostly includes examples with triple loops.</p>
A4	What are different Optimisation Techniques to speed up code?	<p><b>Algorithmic optimisation</b></p> <ul style="list-style-type: none"> <li>• Generally, it is the first thing to look for.</li> <li>• Are there any ways to compute less?</li> <li>• Can we use an algorithm with different <math>O(N)</math>?</li> </ul> <p><b>Let NumPy/pandas do the work</b></p> <p><b>Memoisation (caching)</b> - do not compute the same thing twice.</p> <ul style="list-style-type: none"> <li>• Replace as much of your code with third-party libraries - they are likely better optimised than what you had time for.</li> </ul>

		<ul style="list-style-type: none"> <li>• Are you using all the cores in your system? Following step 1 above first will likely mean this is true. If you still see only one core being used, you need to parallelise your program.</li> <li>• You could transfer the hot spots, i.e. the few lines or functions in which most of the time is spent, to compiled code (numba or Cython).</li> </ul>
A5	What is the general Optimisation Guideline?	<ol style="list-style-type: none"> <li><b>1. Convince yourself that your program is too slow.</b> Does it hurt if the program runs slowly? Is it slow for a realistic use case? Are there any external causes (network, IO, ...) Find bottlenecks by profiling - measure, don't guess! One accurate measurement is worth a thousand expert opinions. - Grace Hopper</li> <li><b>2. Optimisation costs.</b> Optimisation is time-consuming. How much readability and maintainability will suffer? Is it cheaper to use better hardware (hardware is cheaper than the developer time)?</li> <li><b>3. Do not optimize while you develop</b> Correct first, fast later.</li> <li><b>4. Always check the result of optimisation</b> (unit tests, coverage)</li> </ol>
A6	What is the difference between a tuple and a list and when do I use a tuple?	Tuples are the immutable equivalent of lists. Tuples are fast at finding their elements. They tend to use slightly less memory than a list. If you know that something should never change, then use a tuple.
A7	What can I do to avoid bugs?	<ul style="list-style-type: none"> <li>• Write your code with testing and debugging in mind.</li> <li>• Keep It Simple, Stupid (KISS): <ul style="list-style-type: none"> <li>◦ What is the simplest thing that could possibly work?</li> </ul> </li> <li>• Don't Repeat Yourself (DRY): <ul style="list-style-type: none"> <li>◦ Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.</li> <li>◦ Constants, algorithms, etc.</li> </ul> </li> <li>• Try to limit interdependencies of your code (Loose Coupling).</li> <li>• Give your variables, functions, and modules meaningful names (not cryptic/mathematical names).</li> </ul>
A8	What are generators?	Generators are useful to create long sequences without using up the memory to store all the values. In a short syntax, they let you create iterators to use elsewhere.
A9	What are classes and why do we use them?	Classes allow you to create your own holders for data, plus functions (class methods) which you need to modify it. Classes (and class objects) are not a required part of a programming language, and many languages using other programming paradigms (for example, purely procedural languages such as C) have existed for a long time. However, an object-oriented approach works well at avoiding coding mistakes when combining structured collections of data and functions modifying or processing that data. It does this by binding the data and the method together, making it much harder to apply them to the wrong thing.
A10	Between which things do you need to balance when it comes to code quality?	<ul style="list-style-type: none"> <li>• <b>Maintainability:</b> easy &amp; understandable.</li> <li>• <b>Performance:</b> fast &amp; secure.</li> <li>• <b>Resources:</b> size of the machine &amp; developer time.</li> </ul>

**Questions of Type B** (some questions were asked two times in different formulations to simulate students referring to questions with different wording and showing the system is robust enough to recognize the underlying exercise either way). Questions with shorter and longer answers were chosen.

ID	Question	Expected Answer
B1	Write function bin2dec(x2) which takes a positive binary number $x_2 = a_n a_{n-1} \dots a_1 a_0 \dots . \dots b_1 b_2 b_3 \dots$ as a string (e.g. '1011.101'), computes its decimal equivalent $x_{10}$ , and returns it as a float. Input string does not have to contain fractional part - radix point could be missing.	<pre>def bin2dec(x2):     if '.' in x2:         integer2, fractional2 = x2.split('.')     else:         integer2, fractional2 = x2, '0'     integer10 = sum(int(digit) * 2**i for i, digit in enumerate(reversed(integer2)))     fractional10 = sum(int(digit) * 2**(-i) for i, digit in enumerate(fractional2, start=1))     return integer10 + fractional10</pre> <pre>assert bin2dec('110101') == 53 assert abs(bin2dec('.01101') - 0.40625) &lt; 1e-10 assert abs(bin2dec('110101.01101') - 53.40625) &lt; 1e-10</pre>
B2	help me write a function bin2dec(x2) that converts a binary string (e.g., '1011.101') to its decimal equivalent as a float? The input might not always include a fractional part.	See above cell
B3	Let us assume we have a machine that represents floating-point numbers using the following representation: $x = \pm(0.b_1b_2b_3)2^{\times 2e, \text{quad}b_1, b_2, b_3, \text{ein}0, 1} x = \pm(0.b_1b_2b_3)2^{\times 2^e}$ , $\text{quad} b_1, b_2, b_3, e$ $\text{in} \{0, 1\}$ $x = \pm(0.b_1b_2b_3)2^{\times 2e, \text{quad}b_1, b_2, b_3, \text{ein}0, 1}$ . How many machine numbers are there? What are they? 2. How do the machine numbers change if we enforce normalization?	<pre>import itertools def compute_decimal(s, b1, b2, b3, e):     mantissa = b1 * 2**-1 + b2 * 2**-2 + b3 * 2**-3     return (-1)s * mantissa * 2e machine_numbers = [compute_decimal(bits) for bits in itertools.product([(0, 1) for _ in range(5)])] print(f'Machine numbers: {machine_numbers}')</pre> <p>We can see there are duplicates. Let us show and count only unique numbers:</p> <pre>machine_numbers = set(machine_numbers) print(f'Machine numbers: {machine_numbers}')</pre> <pre>print(f'Number of machine numbers: {len(machine_numbers)}')</pre> <pre>def compute_decimal_normalised(s, b2, b3, e):     mantissa = 1 * 2**-1 + b2 * 2**-2 + b3 * 2**-3     return (-1)s * mantissa * 2e machine_numbers = [compute_decimal_normalised(bits) for bits in itertools.product([(0, 1) for _ in range(4)])] machine_numbers = set(machine_numbers) print(f'Machine numbers: {machine_numbers}')</pre> <pre>print(f'Number of machine numbers: {len(machine_numbers)}')</pre>
B4	I have this representation $x = \pm(0.b_1b_2b_3)2^{\times 2e, \text{quad}b_1, b_2, b_3, \text{ein}0, 1} x = \pm(0.b_1b_2b_3)2^{\times 2^e}$ , $\text{quad} b_1, b_2, b_3, e$ $\text{in} \{0, 1\}$ $x = \pm(0.b_1b_2b_3)2^{\times 2e, \text{quad}b_1, b_2, b_3, \text{ein}0, 1}$	See above cell

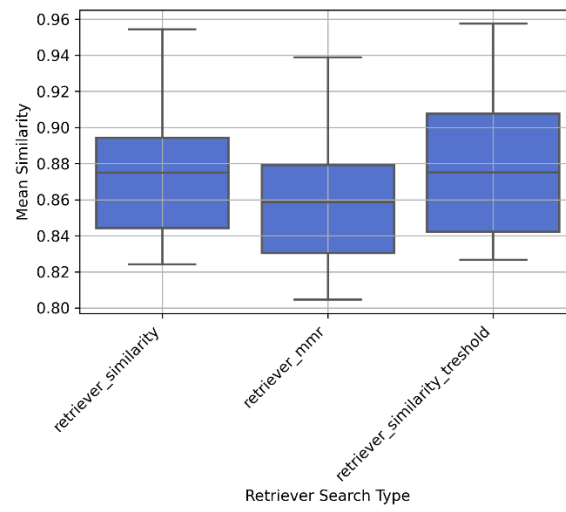
	<p>×2e,quad b1,b2,b3,ein 0,1. How do I find out how many machine numbers there are and what they are? and how do they change if we enforce normalization?</p>	
B5	<p>Determine machine representation of the decimal number -24.98746 in both single and double precision.</p>	<p>Let us begin by defining a function dec2bin to help us convert decimal numbers into binary.</p> <pre>def dec2bin(x10, ndecimal=10):     integer10, fractional10 = divmod(x10, 1)     integer = fractional10 == 0     integer2 = []     while integer10 &gt; 0:         integer10, remainder = divmod(integer10, 2)         integer2.append(int(remainder))     integer2 = ''.join(map(str, reversed(integer2)))     fractional2 = []     while True:         b, fractional10 = divmod(2*fractional10, 1)         fractional2.append(int(b))         if len(fractional2) == ndecimal:             break     fractional2 = ''.join(map(str, fractional2))     return integer2 if integer else integer2 + '.' + fractional2 assert dec2bin(118.61328125, ndecimal=8) == '1110110.10011101' assert dec2bin(118) == '1110110' assert dec2bin(0.61328125, ndecimal=8) == '.10011101'</pre> <p>We need to determine: * sign * mantissa * exponent</p> <p>1. Single precision The formula is:  <math>x = (-1)^s \times 2^{e-127} \times (1 + 0.f)_2</math>  <math>x = (-1)^s \times 2^{e-127} \times (1 + 0.f)^2</math></p> <p>Since the number -24.98746 is negative, sign bit <math>s</math> is 1, so that <math>(-1)^s = -1</math>. In the next step, we convert the number to its binary representation. For that, we can use dec2bin function: dec2bin(24.98746, ndecimal=30) The binary number we obtained is not normalized (please remember that a normalized number is in 1.xxxx form). Therefore, we shift the (radix) point 4 times to the left and the number we get is:</p> $1.1000111111001100101000101101101101 \times 2^4 = 1.1000111111001100101000101101101101 \times 2^4$ <p>If we compare this number to the formula for single precision: <math>x = (-1)^s \times 2^{e-127} \times (1 + 0.f)_2</math> we can see that: <math>2^{e-127} = 2^4 \Rightarrow e-127 = 4 \Rightarrow e = 131</math></p> <p>Now, we need to convert 131 to its binary form to be able to place it in computer memory: dec2bin(131) Finally, we select 23 bits after 1. to save as the mantissa: 10001111110011001010001. And that's it :) Single precision representation of our number is sign + exponent + mantissa: 1100000111000111110011001010001 The procedure to do double precision is the same sign = '1' exponent = '10000011' mantissa = '10001111110011001010001' single_precision =</p>

		<p>sign + exponent + mantissa print(single_precision) I will let you do the double precision. The process is the same, and the final solution is:</p> <pre>110000000011100011111100110010100010110110110110000101110110000</pre> <p><b>Hint:</b> You may find this online tool helpful: * <a href="https://www.h-schmidt.net/FloatConverter/IEEE754.html">https://www.h-schmidt.net/FloatConverter/IEEE754.html</a> * <a href="https://babbage.cs.qc.cuny.edu/ieee-754.old/decimal.html">https://babbage.cs.qc.cuny.edu/ieee-754.old/decimal.html</a></p>
B6	<p>What decimal floating-point number corresponds to \$(1 01010111 10011010001001000110100)_2\$ according to IEEE-754?</p>	<p>this number in binary is:</p> <pre>1 01010111 10011010001001000110100 *</pre> <p>The first bit (1) is the sign: \$s = 1\$  Bits 2-9 (01010111) are the exponent: \$e = 87\$  Remaining 23 bits (10011010001001000110100) are part of the mantissa: \$m = (1.10011010001001000110100)_2 = 1.60211801528930664063\$  Finally, the number we are looking for is:  <math>(-1)^s \times m \times 2^{e-127}</math>  <math>s = 1</math>  <math>m = \text{bin2dec}('1.10011010001001000110100')</math>  <math>e = \text{bin2dec}('01010111')</math>  <math>x = (-1)^s \times m \times 2^{(e-127)}</math>  print(f'The number with representation 1 01010111 10011010001001000110100 is {x}.')</p>
B7	<p>By applying the theorem of loss of precision, compute how many significant digits are lost in subtraction \$x-y\$.</p> <pre>x=0.8796421358quad y=0.8795374261x = 0.8796421358quad y = 0.8795374261x=0.8796421358quad y=0.8795374261</pre>	<pre>x = 0.8796421358 y = 0.8795374261 r = 1 - y/x print(f'{r=}') By computing \log_2 import numpy as np log2r = np.log2(r) print(f'{log2r=}') we can see that \$r\$ lies between \$2^{-13}\$ and \$2^{-14}\$: \$2^{-14} &lt; r &lt; 2^{-13}\$ Therefore, we can conclude that in the subtraction we lost at least 13, and at most 14 significant digits.</pre>
B8	<p>how many significant digits are lost in subtraction \$x-y\$.</p> <pre>x=0.8796421358quad y=0.8795374261x = 0.8796421358quad y = 0.8795374261x=0.8796421358quad y=0.8795374261</pre>	<p>See cell above</p>
B9	<p>For what values of \$x\$, could the loss of significance by subtraction occur. What could you do to prevent this? 1. \$f(x) = \cos^2(x) - \sin^2(x)\$ 2. \$f(x) = \ln(x) - 1\$</p>	<p>1. Problem for \$x \approx \pi/4\$.  <math>f(x)</math> can be computed as  <math>\cos^2 x = \cos^2(x) - \sin^2(x)</math>  to avoid subtraction.  2. Problem for \$x \approx e\$,  <math>f(x) = \ln(x) - 1 = \ln(x) - \ln(e) = \ln(\frac{x}{e})</math>  to avoid subtraction.</p>
B10	<p>Use base to check if array q is a copy of array p. Now, do the same for a simple indexed slice of p (e.g. p[1:2,3:4]).</p>	<pre>p = np.array([[0, 1, 2],[3, 4, 5],[6, 7, 8],[9, 10, 11]]) rows = [0,0,3,3] # indices for rows cols = [0,2,0,2] # indices for columns q=p[rows,cols] print(q.base is p) s = p[1:2,3:4] print(s.base is p)</pre>



## Appendix -B

Impact of Different Search Types on Retriever Performance Across 10 Questions of Type A



## References

all-MiniLM-L6-v2, n.d. [Online]

Available at: <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>

BAAI/bge-base-en-v1.5, n.d. [Online]

Available at: <https://huggingface.co/BAAI/bge-base-en-v1.5>

Barshay, J., 2020. *Reframing ed tech to save teachers time and reduce workloads*, s.l.: <https://hechingerreport.org/ai-in-education-reframing-ed-tech-to-save-teachers-time-and-reduce-workloads/>.

Choi, N., 2023. *GitHub Blog*. [Online]

Available at: <https://github.blog/2023-10-30-the-architecture-of-todays-llm-applications/>

Chroma, 2024. [Online]

Available at: <https://www.trychroma.com/>

Dempere, J., Modugu, K., Hesham, . A. & Ramasamy, L. K., 2023. *The impact of ChatGPT on higher education*. [Online]

Available at: <https://www.frontiersin.org/articles/10.3389/feduc.2023.1206936/full>

Hattie, J. & Timperley, H., 2007. The Power of Feedback. *Review of Educational Research*, 77(81).

Hugging Face, S., 2024. [Online]

Available at: <https://huggingface.co/sentence-transformers>

Jones, K. O., Harland, J., Reid, J. M. & Bartlett, R., 2009. Relationship between examination questions and bloom's taxonomy. *IEEE*, Issue 39.

Jupyter, 2024. [Online]

Available at: <https://jupyter.org/>

Ka Yuk Chan, C. & Hu, . W., 2023. Students' voices on generative AI: perceptions, benefits, and challenges in higher education. *International Journal of Educational Technology in Higher Education*, p. 18.

Langchain, n.d. [Online]

Available at:

[https://api.python.langchain.com/en/latest/embeddings/langchain\\_community\\_embeddings.fastembed.FastEmbedEmbeddings.html](https://api.python.langchain.com/en/latest/embeddings/langchain_community_embeddings.fastembed.FastEmbedEmbeddings.html)

Langchain, n.d. [Online]

Available at: <https://www.langchain.com/>

Leddo, J. & Garg, K., 2021. COMPARING THE EFFECTIVENESS OF AI-POWERED EDUCATIONAL SOFTWARE TO HUMAN TEACHERS. *International Journal of Social Science and Economic Research*, p. 11.

Lehto, T., 2024. *Developing LLM-powered Applications Using Modern Frameworks*. [Online]

Available at: <https://www.theseus.fi/handle/10024/862271>

Martin, L., 2024. *GitHub*. [Online]

Available at: [https://github.com/langchain-ai/rag-from-scratch/blob/main/rag\\_from\\_scratch\\_1\\_to\\_4.ipynb](https://github.com/langchain-ai/rag-from-scratch/blob/main/rag_from_scratch_1_to_4.ipynb)

Maryamah, M. et al., 2024. Chatbots in Academia: A Retrieval-Augmented Generation Approach for Improved Efficient Information Access. *IEEE*, Issue 16th International Conference on Knowledge and Smart Technology (KST), p. 6.

msmarco-distilbert-base-tas-b, n.d. [Online]

Available at: <https://huggingface.co/sentence-transformers/msmarco-distilbert-base-tas-b>

Neumann, M., Rauschenberge, M. & Schoen, . E.-M., 2023. “We Need To Talk About ChatGPT”: The Future of AI and Higher Education. *IEEE/ACM*, Issue 5th International Workshop on Software Engineering Education for the Next Generation (SEENG), p. 4.

OpenAI, n.d. *Chat GPT*. [Online]

Available at: <https://openai.com/chatgpt/>

React, 2024. [Online]

Available at: <https://react.dev/>

Steenbergen-Hu, S. & Cooper, H., 2014. A Meta-Analysis of the Effectiveness of Intelligent Tutoring Systems on College Students’ Academic Learning. *Journal of Educational Psychology*, p. 17.

stsb-roberta-base-v2, n.d. [Online]

Available at: <https://huggingface.co/sentence-transformers/stsb-roberta-base-v2>

Tajik, E. & Tajik, . F., 2023. *A comprehensive Examination of the potential application of Chat GPT in Higher Education Institutions*. [Online]

Available at: [https://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=4699304](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4699304)

VanLehn, K., 2006. The Behavior of Tutoring Systems. *International Journal of Artificial Intelligence in Education*.