Article

Not peer-reviewed version

LLM Intelligent Agent Tutoring in Higher Education Courses using a RAG Approach

Horia Modran * , Ioana Corina Bogdan , Doru Ursuțiu , Cornel Samoila , Paul Livius Modran

Posted Date: 5 July 2024

doi: 10.20944/preprints202407.0519.v1

Keywords: AI; Chatbot; LLM; RAG; Education

Article

LLM Intelligent Agent Tutoring in Higher Education Courses Using a RAG Approach

Horia Alexandru Modran [1], Ioana Corina Bogdan [1], Doru Ursuțiu [1,2], Cornel Samoilă [1,3] and Paul Livius Modran [1,4]

[1] Transilvania University of Brașov, Brașov, Romania; horia.modran@unitbv.ro, corina.bogdan@unitbv.ro, udoru@unitbv.ro, csam@unitbv.ro, modranp@gmail.com
[2] Romanian Academy of Scientists, Bucharest, Romania
[3] Romanian Academy of Technical Sciences, Bucharest, Romania
[4] Mass Mutual Romania, Bucharest, Romania

Abstract. Conventional tutoring approaches are confronted with limitations such as restricted availability, inconsistent pedagogical quality, and scalability constraints. Furthermore, the exclusive use of Large Language Models (LLMs) like ChatGPT in education has its shortcomings, including the potential for incorrect responses and the lack of customized guidance aligned with specific course content. This research proposes an innovative intelligent chatbot tutoring system, integrating the Retrieval Augmented Generation (RAG) approach with a custom LLM. The developed system aims to overcome the limitations of traditional tutoring and general-purpose LLMs by providing accurate, contextually relevant, and personalized assistance, thereby enhancing student understanding and engagement. The system, powered by an intelligent agent, retrieves relevant information from curated academic sources, incorporates interactive features for user feedback, and utilizes machine learning algorithms for ongoing performance enhancement, ensuring a robust and effective tutoring experience. The anticipated outcome is an enriched educational experience for university students, advancement in personalized learning, and improved student engagement, retention, and academic performance. Through continuous research and refinement, it is expected that the intelligent chatbot tutor will assume an important role in enhancing and supporting the educational journey of students.

Keywords: AI; Chatbot; LLM; RAG; Education

1. Introduction

Traditional methods of student tutoring often face challenges such as limited availability, inconsistency in teaching quality, and scalability issues. Moreover, relying solely on general-purpose language models (LLMs) like ChatGPT for educational purposes poses several drawbacks, including inaccurate responses and the inability to provide tailored guidance based on specific course content. Taking into consideration these challenges, there is a pressing need to develop innovative solutions that leverage the power of Artificial Intelligence (AI) to enhance the learning experience for students.

A comprehensive review of relevant research articles was conducted to explore the state of art in the fields of Retrival Augumented Generation (RAG) in Large Language Models (LLMs), Intelligent ChatBot Tutoring, and their application into Higer Education. This review emphasizes the variety of strategies by which RAG can be utilized to overcome the inherent constraints of general-purpose LLMs when performing tasks that require comprehensive knowledge [1].

The review conducted by M. Ashfaque et. al. [2] delves into the integration of AI and NLP in intelligent tutoring systems, with a focus on Chatbots as virtual tutors. It discusses the diverse applications of Chatbots in education and various sectors, highlighting the need for continuous improvement in Chatbot design and development for enhanced functionality and user experience.

Research paper [3] investigates tensions arising from the integration of large language model-based chatbots in higher education, emphasizing the need for clear guidelines and collaborative rule-making. Students and teachers express concerns about the quality of learning, the value of diplomas, and the evolving roles in education due to LLM-based chatbots. The study highlights the importance of understanding the changing human-technology relationship and the implications for learning objectives and division of labor within educational settings.

The paper [4] presents a hybrid model integrating Large Language Models (LLMs) and Chatbots for efficient access to educational materials in higher education. It emphasizes the importance of prompt engineering and content knowledge in maximizing the potential of LLM. Practical implementations, such as question-answering chatbots, demonstrate the effectiveness of the proposed programming environment. Another research [5] highlights the transformative capabilities of LLMs in education and underscores the role of critical thinking and iterative design for optimal performance.

The article [6] outlines the creation of TutorBot+, a chatGPT-based feedback tool for programming students at Universidad Católica de la Santísima Concepción, aiming to enhance learning and computational thinking skills. Preliminary results show successful integration with an LMS and potential for improving the educational experience. N. Bakas et. al. [7] proposes using ChatGPT's API for interactive tutoring, highlighting the transformative impact of LLMs in education by enabling dynamic, customized learning experiences.

The study on Student Interaction with NewtBot [8] demonstrates positive student perceptions of using generative AI for schoolwork and the effectiveness of internal prompt engineering in customizing LLM chatbot behaviors for improved academic engagement. S. Siriwardhan et. al. [9] introduces RAG-end2end for domain adaptation in ODQA, enhancing RAG models with auxiliary signals for improved factual consistency and reduced hallucinations. Results demonstrate the effectiveness of joint retriever-generator training in specialized domains, suggesting potential applications in educational chatbot tutoring systems.

In prior research, the same group of researchers proposed an instructional methodology for training engineers in all necessary procedures for the creation, validation, and implementation of machine learning-based systems [10], as well as strategies for incorporating Artificial Intelligence and ChatGPT into Higher Engineering Education [11]. The current study seeks to transcend the limitations of traditional tutoring and general-purpose LLMs by providing accurate, contextually relevant, and personalized assistance, thereby enhancing student understanding and engagement. This system, guided by an intelligent agent and utilizing a RAG approach, collects relevant information from meticulously curated academic sources, incorporates interactive components for user feedback, and applies machine learning methods for ongoing performance optimization, ensuring a proficient tutoring experience.

2. LLM Intelligent Chatbot with RAG

The landscape of higher education is evolving rapidly with the integration of advanced technologies, particularly AI-driven solutions. Large Language Models (LLMs) have emerged as a powerful tool, capable of processing and generating human-like text, making them suitable for a variety of applications, including tutoring. However, when deployed in their raw form, these models exhibit limitations such as inaccuracies and a lack of domain-specific knowledge. To address these issues, the integration of LLMs with Retrieval-Augmented Generation (RAG) techniques offers a promising field for developing intelligent chatbot tutors that provide precise, contextually relevant, and personalized educational support.

RAG leverages the strengths of both retrieval-based and generation-based methods. In this approach, a retrieval component first searches for relevant information from a curated set of academic resources, ensuring the content is accurate and domain-specific. The generation component then synthesizes this information to produce coherent and contextually appropriate responses. This hybrid model not only mitigates the inaccuracies often associated with standalone LLMs but also enhances their ability to provide tailored guidance aligned with specific course content.

Implementing a RAG-based intelligent chatbot for higher education involves several key steps. Firstly, a comprehensive database of educational materials, including textbooks, research articles, lecture notes, and other relevant resources, must be curated. This database serves as the foundation for the retrieval component, ensuring that the information fed into the LLM is both accurate and contextually relevant. Next, the chatbot is trained using prompt engineering techniques to fine-tune its responses, ensuring they are aligned with educational objectives and learning outcomes.

Interactive components are crucial for effective tutoring. A RAG-based chatbot can incorporate features such as quizzes, practice problems, and instant feedback mechanisms. These components not only engage students actively but also help in reinforcing their understanding of the material. Additionally, the chatbot can adapt to individual learning paces and styles, providing a personalized learning experience. By analyzing student interactions and feedback, the system can continuously improve its performance through iterative design and machine learning techniques.

Practical implementations of such systems demonstrate their potential to transform the educational landscape. For instance, question-answering chatbots developed using RAG approaches have shown significant improvements in delivering precise and comprehensive answers compared to traditional LLMs. These chatbots can assist students with homework, clarify complex concepts, and even offer guidance on research projects, thereby enhancing the overall learning experience.

Moreover, the integration of RAG-based chatbots in Learning Management Systems (LMS) provides a seamless and accessible tutoring solution. Students can interact with the chatbot anytime, ensuring consistent support outside the classroom. This continuous availability helps address the issue of limited tutoring resources and offers scalable solutions for educational institutions.

3. RAG Agent and VectorStore

The LlamaIndex framework was employed to enhance the Large Language Model (LLM) using a Retrieval Augmented Generation (RAG) approach. LlamaIndex is a data framework designed to optimize the integration of RAG with an LLM and it is constructed upon the foundation of the OpenAI API. This framework enhances the efficiency and accuracy of information retrieval and generation processes, making it highly suitable for applications in intelligent chatbot tutoring. LlamaIndex stands out due to its ability to manage vast amounts of data, facilitate efficient vector storage, and enable seamless interaction between the retrieval and generation components of RAG systems [11,12].

LlamaIndex provides a robust infrastructure for indexing and retrieving large datasets. It employs advanced indexing techniques that allow for quick and precise access to relevant information. One of the primary advantages of this framework is its capability to handle high-dimensional data vectors, which are essential for representing the rich semantic information contained in educational materials. By optimizing the storage and retrieval processes, it ensures that the RAG system can operate at peak performance, delivering accurate and contextually relevant responses to student queries.

A RAG agent operates by first retrieving pertinent information from a pre-defined knowledge base and then generating a coherent and contextually appropriate response. The knowledge base is indexed using LlamaIndex, which transforms the educational content into high-dimensional vectors stored in a vector database. This approach facilitates rapid and efficient retrieval of information, significantly enhancing the performance of the LLM.

The workflow of the agent is illustrated in Figure 1. It begins with a student's query, which is encoded into a query vector and sent to the Router Query Engine. LlamaIndex then performs a similarity search within the vector store to find the most relevant documents or data points. The retrieval component ensures that the information pulled is accurate and aligned with the educational context of the query. This retrieved information is then fed into the LLM, which generates a comprehensive and tailored response.

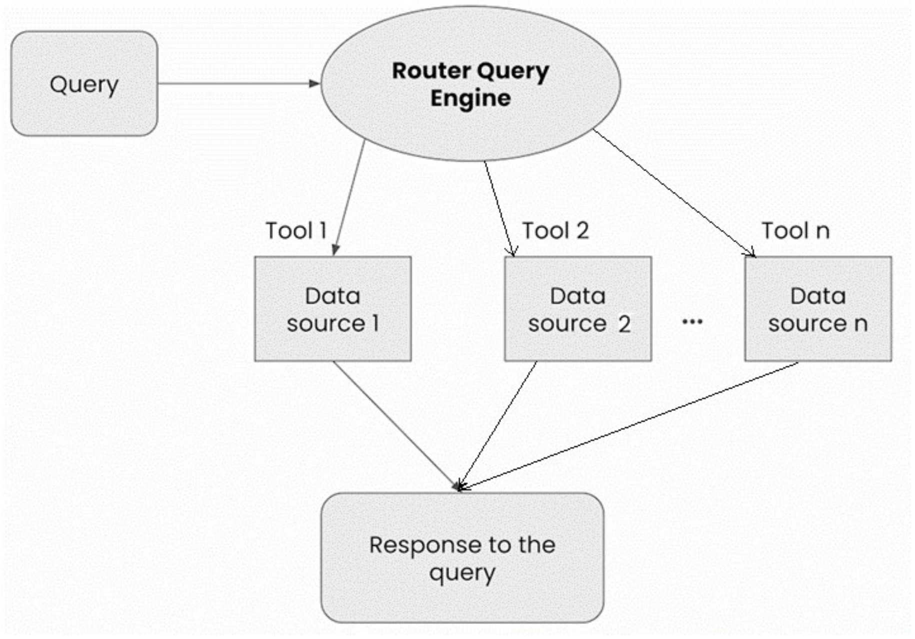


Figure 1. RAG Agent Architecture.

The vector database is a critical component in the RAG framework, acting as a repository for the high-dimensional vectors representing the indexed educational content. LlamaIndex leverages state-of-the-art algorithms to create and manage this vector store, ensuring that the vectors are organized in a manner that allows for efficient retrieval. This setup not only speeds up the search process but also enhances the accuracy of the retrieved information by reducing the likelihood of retrieving irrelevant or incorrect data.

By indexing a comprehensive knowledge base, the system ensures that the information retrieved is always relevant and reliable, thereby enhancing the quality of tutoring. The use of the vector store allows the system to scale effortlessly, accommodating a growing repository of educational materials without sacrificing performance. This scalability is particularly beneficial for educational institutions looking to implement AI-driven tutoring solutions across multiple courses and disciplines.

4. Chatbot Features and Functionalities

4.1. Processing the Documents and Creating the Vector Store

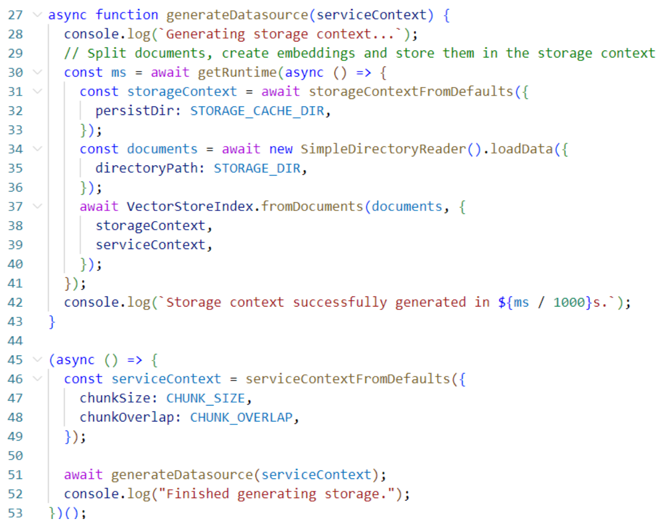
This section details the development steps and core functionalities of the intelligent Chatbot agent developed for higher education. It covers the process of persisting data in a vector store using the LlamaIndex library, the methods employed to load and index educational materials, and the specific configurations set to optimize data retrieval and response generation. Through this detailed exposition, the current section presents how the system efficiently manages large volumes of educational content and ensures accurate, contextually relevant interactions with students, thereby enhancing the overall tutoring experience.

A key feature of the intelligent chatbot tutoring system is its ability to persist data in a vector store, which is facilitated using the LlamaIndex library. This capability ensures that the chatbot can efficiently manage and retrieve large volumes of educational content, providing accurate and relevant responses to student queries.

The process of creating the Vector Store involves several steps. Initially, a Storage content is created using the *storageContextFromDefaults()* function. This function sets up the default storage context, which serves as the foundation for the Vector Store. During the indexing process, a serviceContext is created to manage the configuration and execution of the indexing tasks. Within this context, specific parameters such as chunk size and chunk overlap are set to optimize the indexing process. The use of chunk size and overlap parameters is critical for balancing the granularity and coherence of the indexed content. A chunk size of 512 ensures that each segment of the document is of manageable length, allowing the model to capture sufficient context within each vector. The chunk overlap of 20 provides a slight overlap between consecutive chunks, preserving the continuity of information and improving the system's ability to retrieve contextually relevant data.

Following the creation of the Storage content, documents are loaded into the system using the *SimpleDirectoryReader().loadData()* function. This function is designed to handle entire directories, allowing for the bulk import of educational content. By passing the directory containing the educational materials as a parameter, the function reads and processes the files, converting them into a format suitable for indexing. The loaded documents are then used to create a VectorStoreIndex using the *VectorStoreIndex.fromDocuments()* function. This function transforms the loaded documents into a format suitable for storage in the vector store. These vectors encapsulate the semantic information contained in the documents, enabling efficient and accurate retrieval based on the content's meaning rather than just keyword matching.

The TypeScript implementation responsible for the instantiation and local persistence of the vector store is presented in Figure 2.



```
async function generateDatasource(serviceContext) {
  console.log(`Generating storage context...`);
  // Split documents, create embeddings and store them in the storage context
  const ms = await getRuntime(async () => {
    const storageContext = await storageContextFromDefaults({
      persistDir: STORAGE_CACHE_DIR,
    });
    const documents = await new SimpleDirectoryReader().loadData({
      directoryPath: STORAGE_DIR,
    });
    await VectorStoreIndex.fromDocuments(documents, {
      storageContext,
      serviceContext,
    });
  });
  console.log(`Storage context successfully generated in ${ms / 1000}s.`);
}

(async () => {
  const serviceContext = serviceContextFromDefaults({
    chunkSize: CHUNK_SIZE,
    chunkOverlap: CHUNK_OVERLAP,
  });

  await generateDatasource(serviceContext);
  console.log("Finished generating storage.");
})();
```

Figure 2. Typescript code for creating the Vector Store.

The complete process for handling the documents and creating the local vector store is illustrated in Figure 3. The local vector store consists of three key files, each serving a distinct purpose:

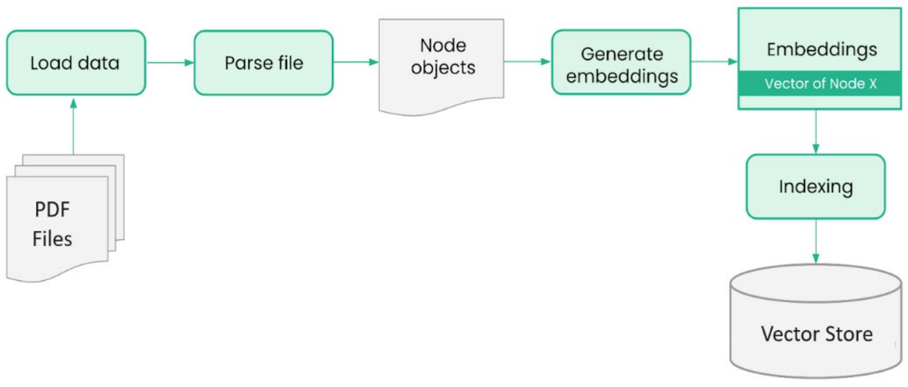


Figure 3. Parsing data and creating the vector store.

- *doc_store.json*—this file contains the raw documents that have been loaded into the system. It serves as a repository of the original educational content, preserving the text and metadata associated with each document.

- *index_store.json*—it maintains the indexing information for the documents stored in the system. It includes the structures and mappings that allow for efficient searching and retrieval of documents based on their content.
- *vector_store.json* stores the high-dimensional vectors generated from the indexed documents. These vectors are created using an embedding technique, and they capture the semantic meaning of the documents. They are used for tasks such as similarity search and clustering.

4.2. Utilizing RAG to Augment the LLM

In this section, the integration of the Retrieval-Augmented Generation (RAG) model with the Language Model (LLM) is explored to enhance its capabilities. Specifically, the LlamaIndex library is utilized in conjunction with the OpenAI GPT-4 model. The objective is to augment the LLM using RAG, leveraging the vector store that was previously created.

A TypeScript web application has been developed with both front-end and back-end components. This application can engage in real-time chat with the data and streaming the response. The process involves several steps, which are detailed below:

1. Creation of the Chat Engine—this first step involves the creation of a chat engine using the *createChatEngine()* function. This engine is built with an OpenAI GPT-4 LLM, which forms the backbone of the chat system.
2. User Message Processing—upon receiving a user message, the content is converted into a format that is compatible with the LlamaIndex and OpenAI. This conversion is crucial for ensuring that the input is appropriately structured for both the retrieval and generation processes. The formatted message serves as the basis for querying the vector store and generating relevant responses.
3. Retrieving and generating responses—the core functionality of the chatbot is realized through the invocation of LlamaIndex's *chatEngine.chat()* function. This method leverages the RAG approach by first retrieving relevant information from the vector store based on the user's query. The retrieved information is then used to generate a coherent and contextually appropriate response using the GPT-4 model. This function is designed to stream the response in real-time, enhancing the interactivity of the Chatbot.
4. Streaming the response—the stream generated by the chat engine is consumed by the front-end client. This allows for real-time interaction with the user, providing them with immediate feedback and responses.
5. Piping the LlamaIndexStream to Response—finally, the LlamaIndexStream is piped to the response using the *stream.pipeThrough()* function. The use of this method allows for efficient handling of the streamed data, maintaining the integrity and coherence of the response as it is displayed to the user.

The complete workflow of the application is depicted in Figure 4. This approach allows the power of RAG and LLM to be leveraged in a seamless manner, providing a robust and responsive chat system. By integrating these advanced models with the vector store, highly accurate and contextually relevant responses can be delivered in real time. This marks a significant advancement in the field of AI-powered chat systems, opening new possibilities for user interaction and engagement.

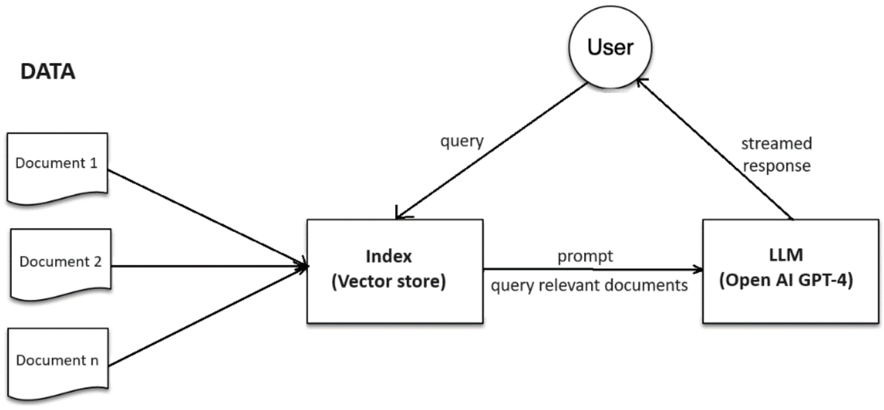
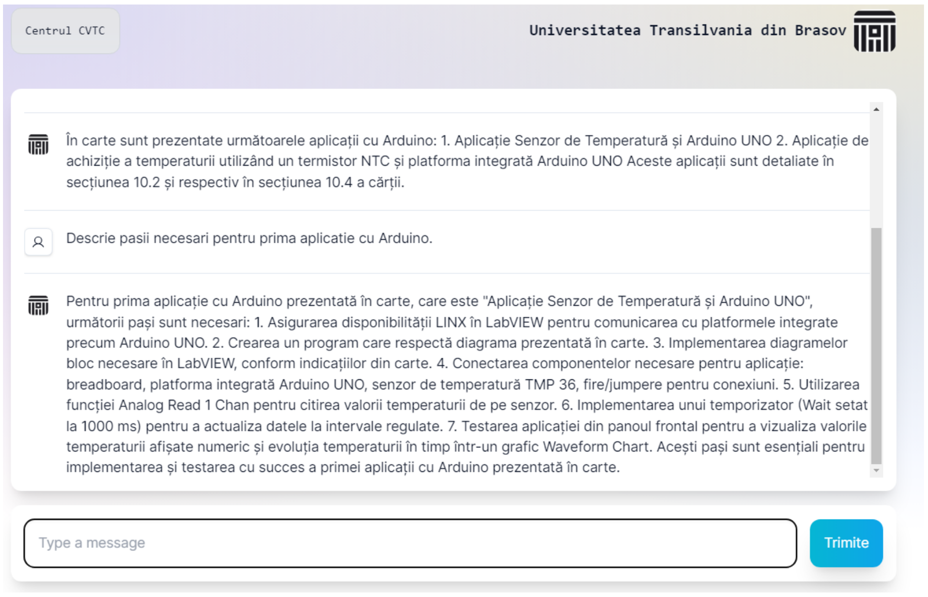


Figure 4. Application workflow.

The application is currently in the testing phase, incorporating educational materials relevant to the Virtual Instrumentation course at the Transilvania University of Brasov. To construct the knowledge base for the application, two primary documents were employed:

- Virtual Instrumentation: Laboratory Guide [14]—this document provides comprehensive laboratory exercises and practical guidance on virtual instrumentation, serving as a foundational resource for hands-on learning and experimentation.
- Introduction to LabVIEW Graphic Programming with Applications in Electronics, Telecommunications, and Information Technologies [15]—it covers fundamental concepts and practical applications in electronics, telecommunications, and IT, making it a critical resource for understanding the software tools used in virtual instrumentation.

To evaluate the effectiveness of the application, it is currently being tested to determine how well it responds to queries related to the documents integrated into its knowledge base. This testing phase involves asking various questions about the content covered in those two documents. Preliminary results have been promising, indicating that the chatbot can accurately retrieve and generate contextually appropriate responses based on these resources. Furthermore, the application stores previous conversations locally, allowing it to maintain memory and context over the course of an interaction. Figure 5 illustrates a sample question regarding the steps needed to build an Arduino Application in LabVIEW, demonstrating the application's capability to handle specific queries effectively.



În carte sunt prezentate următoarele aplicații cu Arduino: 1. Aplicație Senzor de Temperatură și Arduino UNO 2. Aplicație de achiziție a temperaturii utilizând un termistor NTC și platforma integrată Arduino UNO Aceste aplicații sunt detaliate în secțiunea 10.2 și respectiv în secțiunea 10.4 a cărții.

Descrie pasii necesari pentru prima aplicatie cu Arduino.

Pentru prima aplicație cu Arduino prezentată în carte, care este "Aplicație Senzor de Temperatură și Arduino UNO", următorii pași sunt necesari: 1. Asigurarea disponibilității LINX în LabVIEW pentru comunicarea cu platformele integrate precum Arduino UNO. 2. Crearea un program care respectă diagrama prezentată în carte. 3. Implementarea diagramelor bloc necesare în LabVIEW, conform indicațiilor din carte. 4. Conectarea componentelor necesare pentru aplicație: breadboard, platforma integrată Arduino UNO, senzor de temperatură TMP 36, fire/jumpere pentru conexiuni. 5. Utilizarea funcției Analog Read 1 Chan pentru citirea valorii temperaturii de pe senzor. 6. Implementarea unui temporizator (Wait setat la 1000 ms) pentru a actualiza datele la intervale regulate. 7. Testarea aplicației din panoul frontal pentru a vizualiza valorile temperaturii afișate numeric și evoluția temperaturii în timp într-un grafic Waveform Chart. Acești pași sunt esențiali pentru implementarea și testarea cu succes a primei aplicații cu Arduino prezentată în carte.

Figure 5. Sample question and response.

5. Discussions and Conclusions

The proposed intelligent chatbot tutoring system, integrating Retrieval-Augmented Generation (RAG) with a custom Large Language Model (LLM), marks a significant advancement in educational technology. This system effectively addresses the limitations of traditional tutoring methods and standalone LLMs, offering a robust solution for personalized and contextually relevant educational support. By leveraging RAG, the system accurately retrieves domain-specific information from a curated set of academic sources, ensuring high-quality, tailored guidance aligned with specific course content.

One of the primary strengths of this approach is its ability to efficiently manage and retrieve large volumes of educational content. The LlamaIndex framework, used to create and maintain the vector store, ensures that the information retrieval process is both rapid and precise. This capability is critical for maintaining the accuracy and contextual relevance of the Chatbot's responses, which enhances student engagement and comprehension. Additionally, the system's interactive components provide a dynamic and engaging learning experience that can adapt to individual learning paces and styles.

However, few limitations of the system have been identified. The accuracy and effectiveness of the Chatbot relies heavily on the quality and comprehensiveness of the curated academic sources. Any gaps or biases in these sources could potentially affect the quality of the generated responses. Furthermore, while preliminary results are promising, extensive testing across a broader range of subjects and educational contexts is necessary to fully validate the system's efficacy. The system is set to be tested with students enrolled in the Virtual Instrumentation course at Transilvania University of Brașov during the next academic year.

In conclusion, the integration of RAG with LLMs offers a powerful approach to developing intelligent chatbot tutors capable of providing precise, contextually relevant, and personalized educational support. It is expected that this system will have the potential to significantly improve the quality of tutoring available to university students, fostering enhanced engagement, retention, and academic performance. Future developments of this research will focus on expanding the knowledge base, refining the retrieval and generation algorithms, and incorporating more advanced machine learning techniques to further enhance performance.

References

1. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20). Curran Associates Inc., Red Hook, NY, USA, Article 793, 9459–9474, DOI: https://doi.org/10.48550/arXiv.2005.11401 (2020).
2. Ashfaque, M. W., Tharewal, S., Iqhbal, S., Kayte, C. N.: A Review on Techniques, Characteristics and approaches of an intelligent tutoring Chatbot system, 2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC), Aurangabad, India, 2020, pp. 258-262, DOI: https://doi.org/10.1109/ICSIDEMPC49020.2020.9299583.
3. Carbonel, H., Jullien, J.-M.: Emerging tensions around learning with LLM-based chatbots: A CHAT approach, Networked Learning Conference, 14(1), https://journals.aau.dk/index.php/nlc/article/view/8084.
4. Bratić, D., Šapina, M., Jurečić, D., Žiljak Gršić, J.: Centralized Database Access: Transformer Framework and LLM/Chatbot Integration-Based Hybrid Model. Appl. Syst. Innov. 2024, 7, 17, DOI: https://doi.org/10.3390/asi7010017 (2024).
5. Makharia, R. et al.: AI Tutor Enhanced with Prompt Engineering and Deep Knowledge Tracing," 2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), Gwalior, India, 2024, pp. 1-6, DOI: https://doi.org/10.1109/IATMSI60426.2024.10503187.
6. Martinez-Araneda, C., Gutiérrez, M., Maldonado, D., Gómez, P., Segura, A., Vidal-Castro, C.: Designing a Chatbot to support problem-solving in a programming course, INTED2024 Proceedings, pp. 966-975, DOI: https://doi.org/10.21125/inted.2024.0317 (2024).
7. Bakas, N.P., Papadaki, M., Vagianou, E., Christou, I., Chatzichristofis, S.A.: Integrating LLMs in Higher Education, Through Interactive Problem Solving and Tutoring: Algorithmic Approach and Use Cases, In:

Information Systems. EMCIS 2023. Lecture Notes in Business Information Processing, vol 501. Springer, Cham, DOI: https://doi.org/10.1007/978-3-031-56478-9_21.

8. Lieb, A., Goel, T.: Student Interaction with NewtBot: An LLM-as-tutor Chatbot for Secondary Physics Education. In Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '24), Association for Computing Machinery, New York, NY, USA, Article 614, pp. 1–8, DOI: https://doi.org/10.1145/3613905.3647957.
9. Siriwardhana, S., Weerasekera, R., Wen, E., Kaluarachchi, T., Rana, R., Nanayakkara, S.: Improving the Domain Adaptation of Retrieval Augmented Generation (RAG) Models for Open Domain Question Answering. Transactions of the Association for Computational Linguistics 2023; 11 1–17, DOI: https://doi.org/10.1162/tacl_a_00530.
10. Modran, H.A., Ursutiu, D., Samoila, C., Chamunorwa, T.: Learning Methods Based on Artificial Intelligence in Educating Engineers for the New Jobs of the 5th Industrial Revolution. In: Educating Engineers for Future Industrial Revolutions. ICL 2020. Advances in Intelligent Systems and Computing, vol 1329. Springer, Cham, DOI: https://doi.org/10.1007/978-3-030-68201-9_55 (2021)
11. Modran, H.A., Chamunorwa, T., Ursuțiu, D., Samoilă, C.: Integrating Artificial Intelligence and ChatGPT into Higher Engineering Education. In: Towards a Hybrid, Flexible and Socially Engaged Higher Education. ICL 2023. Lecture Notes in Networks and Systems, vol 899. Springer, Cham, DOI: https://doi.org/10.1007/978-3-031-51979-6_52.
12. ChatGPT API Reference, https://platform.openai.com/docs/api-reference/introduction, last accessed 2024/05/25.
13. LLamaIndex Documentation, https://www.llamaindex.ai/, last accessed 2024/05/28.
14. Modran, H. A., Ursuțiu, D.: Instrumentație Virtuală: Îndrumar de laborator, Transilvania University of Brasov Publishing House, ISBN 9786061915460 (2022).
15. Modran, H. A., Bogdan, I. C., Ursuțiu, D., Samoilă, C.: Introducere în Programarea Grafică LabVIEW cu Aplicații în Electronică, Telecomunicații și Tehnologii Informaționale, Transilvania University of Brasov Publishing House, ISBN 978-606-19-1709-9 (2024).