Generative AI Agent for Tutoring using LangChain

Foo Yan Rong (s231651), Chew Zhi Yi Mark (s231490), Lee Kar Jun Lester (s231437)

Technical University of Denmark

Abstract

The diverse needs of learners has sparked a demand for adaptive educational solutions. This research introduces the employment of intelligent agents as tutors. Utilizing the Retrieval-Augmentation Generation technique and vector store memories via Chain of Thought reasoning, we aim to provide personalized tutelage for every individual through the use of Large Language Models. This report details the methodologies, results, and evaluations using LangChain and OpenAI in our implementation. While highlighting the transformative potential of these agents in providing an engaging educational experience, the research contributes to the discourse on Artificial Intelligence's role in transforming education, presenting a novel approach to a dynamic and adaptive tutoring system.

1. INTRODUCTION

In the realm of education, the challenge of accommodating diverse learning styles and paces has spurred the demand for personalized and adaptive learning solutions. Traditional educational methods often fall short of meeting the individual needs of students, prompting the exploration of innovative approaches. Despite the proliferation of standard Q&A bots, these solutions exhibit limitations in engagement and adaptability. However, with the rapid progress of artificial intelligence, particularly the advancements seen in OpenAI's GPT-3.5 Turbo and LangChain, new opportunities arise for the development of intelligent educational tools.

This research project is motivated by the aspiration to revolutionize the educational landscape beyond the confines of conventional question-and-answer bots. The primary objective is to create a dynamic simulation engine using agents that function as tutors, adapting their roles to accommodate the context and individual needs of learners. Leveraging the capabilities of Retrieval-Augmented Generation (RAG) and incorporating vector store memories into prompt templates through Chain of Thought (CoT) reasoning, this project aims to push the boundaries of educational technology. By harnessing the power of the LangChain framework, the goal is to develop agents capable of effectively fulfilling their roles as dynamic and adaptive tutors. This report delves into the methodologies and outcomes of creating such intelligent agents in the pursuit of transforming the educational experience.

2. RELATED WORK

In this section, we delve into a quick exploration of existing works in the academic landscape that relate to our field of study. This aims to position our work within the context of the current state of knowledge.

2.1 Chain-of-Thought Prompting

Research has been done to prove that Large Language Models (LLM) can naturally acquire improved performance in complex reasoning tasks by generating a series of intermediate reasoning steps as exemplars in the prompt. [1] Significant improvements were shown in arithmetic, common sense, and symbolic reasoning tasks through the use of CoT prompting to elicit multi-step reasoning behavior in the models. However, it is recognized that there exists limitations with such a prompting technique such as that there is no assurance of correct reasoning paths, resulting in the possibility of both correct and incorrect answers.

2.2 Violation of Expectation to improve Theory of Mind capabilities

Theory of Mind involves imputing unobservable mental states to others and it has been proven that as a result of incorporating Violation of Expectation, there is a reduction in error in LLM predictions. [2] Violation of Expectation includes making predictions about reality based on past learning and comparing these predictions against reality to learn from the error in the differences. Through a proposed metacognitive prompting framework, this learning process can be implemented in LLMs whereby models generate “thoughts” that are used as context in subsequent inference steps. The authors also recommend the use of Retrieval-Augmented Generation in future work to supply the relevant context to be used.

2.3 Multi-Agent Conversation

Multiple agents can be utilized to solve diverse tasks through the encouragement of divergent thinking and can be used to significantly improve LLMs. Recently, a framework has been proposed to simplify the development of high-performance multi-agent applications through customizable and conversable agents. [3] It supports flexible conversation patterns by offering a unified conversation interface among agents and promotes modularity by dividing tasks among separate agents. Thus, once one has created an intelligent agent tailored to their specific domain or task, they can be used to partake in multiple conversations with other agents in order to result in higher-order thinking and obtain a more sophisticated response.

3. METHODOLOGY

Tutor agents, by definition, are large language models specifically designed to assist with learning and understanding complex topics. They provide personalized instruction and support to students beyond conventional established one-way, teachers-to-students, knowledge vomiting. There are a multitude of implementations to develop a persistent-memory tutor agent for this particular use case. Starting from the choice of framework, there are a multitude of platforms available Auto-gpt, LlamaIndex, FlowiseAI, and LangChain. Even the implementation of the system comes in infinite possibilities such as system prompting, vector store database, and memory embedding. In this particular project, we explored the implementation of a tutor agent through LangChain with OpenAI's GPT–3.5 turbo as the main large language model.

In this implementation of a tutor agent, we utilize a relatively well-established technique, RAG, that integrates both LLM, and the Information Retrieval System (IR) to produce an accurate and relevant answer.

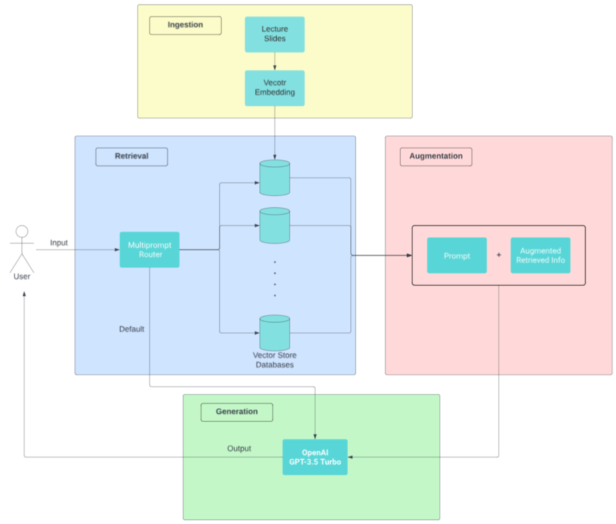

Fig. 1: Data Flow Diagram of Tutor Agent

3.1 Ingestion

Before the RAG pipeline can be commissioned, ingestion is a necessary preparation phase for the information retrieval system to work meaningfully. This stage involves collecting relevant and appropriate data, splitting data into semantically meaningful chunks, generating vector embedding of chunks, and storing the embeddings in a vector store database.

This particular agent will be specifically developed for the use of the Technical University of Denmark's Deep Learning course students. Hence, the data chosen to implement specialized knowledge are in the form of weekly lecture slides that have been divided into various major topics within deep learning. The documents are first loaded using PyPDFLoader where the pdf file is loaded into text format separated by topic within deep learning. Thereafter, the text is divided into semantically sound chunks using RecursiveCharacterTextSplitter with a chunk size of 1500 characters, and an overlap of 500 characters. Subsequently, the OpenAIEmbeddings function is applied to each chunk and stored in the chroma database where each major topic is stored in an individual database.

3.2 Retrieval

In the retrieval stage of the RAG pipeline, documents that had been previously loaded, embedded, and stored in the ingestion phase are identified and retrieved based on the relevance to a query input.

Since deep learning has many major topics that have "neural networks" in their name, we have separated each major topic into a database of its own to prevent cross-contamination of the data. Hence, to retrieve the relevant data from our databases, query input will first be processed by LLM and routed to the most appropriate database using LLMRouterChain. Thereafter, RetrievalQA is used to evaluate the query input for retrieval of any relevant document from the appointed database based on stuff technique. The retrieved data will then be handled in the augmentation phase.

3.3 Augmentation

In the augmentation phase, retrieved data are typically sifted, transformed, verified, and augmented to produce a variety of inputs that are most suited for a particular query leading to better results. It is worth noting that beyond the above steps, it is a common practice to condense and summarize documents retrieved due to the token limit of LLMs. Summarizing information retrieved ensures that repetitive content is omitted and the retrieved data that will be inserted as context to the final LLM is comprehensive and informative for the generation of the replying response.

Since the data is retrieved from lecture slides and has been split into semantically digestible chunks, the most relevant data retrieved is directly evaluated by the LLM in RetrievalQA to produce a summarized context. The context, or augmented data produced is then inserted into the prompt template for the generator LLM that is in the generation stage.

3.4 Generation

Lastly, in the final stage of the pipeline, the LLM synthesizes the augmented data and its understanding of the query through a prompt template to generate a comprehensible and instructive response.

For the final LLM, OpenAI's GPT-3.5 turbo is called using ChatOpenAI with 0 temperature to produce a relatively deterministic response. This response would then be relayed to the user as the tutor agent's response to the query posted. To ensure that the agent responds as per the functional requirement of a tutor agent, prompt engineering is necessary to guide and refine the behavior of the final LLM.

For each major concept taught in Deep Learning, a tutor agent will be responsible for generating a detailed, step-by-step explanation in response to a student's query on that concept. The response is also generated in a way that the agent believes is able to allow the student to understand better. Following the explanation, the agent is tasked with formulating pertinent questions designed to deepen the student's understanding. These questions draw upon the initial inquiry, the student's response, and relevant lecture material, contributing to a comprehensive approach aimed at reinforcing the student's understanding of the underlying concept. An example of an agent tutor's response can be found in Fig. 2 and Fig. 3.

AI: Backpropagation is an algorithm used in neural networks to train the model. It works by adjusting the weights and biases of the network based on the gradient of the loss function. The loss function measures how well the model is performing, and the gradient tells us the direction in which we should update the weights and biases to minimize the loss.

To understand this better, let's imagine you are trying to learn how to ride a bike. At first, you might make some mistakes and lose your balance. But instead of giving up, you learn from those mistakes and make adjustments to improve your balance. Backpropagation works in a similar way for neural networks.

The algorithm calculates the gradient of the loss function with respect to the weights and biases. This gradient tells us how much each weight and bias contributes to the overall loss. By knowing this, we can update the weights and biases in the opposite direction of the gradient, which means we are moving towards minimizing the loss.

Fig. 2: Tutor Agent's Explanation on the Process of Backpropagation upon Student Inquiry.

1. Why is backpropagation important in training a feedforward neural network?
2. How does backpropagation calculate the gradient of the loss function?
3. What happens if we don't use backpropagation to update the weights in a neural network?
4. Can you explain the concept of gradient descent and its relationship with backpropagation?
5. Are there any limitations or challenges associated with backpropagation?

Feel free to ask any of these questions or let me know if you have any other doubts!

Fig. 3: Tutor Agent Generating Follow-up Questions for the Student after Backpropagation Explanation.

4. RESULTS AND EVALUATION

Evaluating the performance of the RAG agent is a pivotal component of this research report as it unravels the insights gained from an in-depth examination of the generative AI model's performance. The evaluation spans both quantitative metrics and qualitative assessments to provide a comprehensive understanding of the model's capabilities and limitations. The following subsections delve into various aspects of the evaluation, including the quality of text generation, scenario-based assessments, user feedback, robustness testing, and benchmarking against baseline models. By exploring these dimensions, we aim to offer a nuanced perspective on the generative AI model's effectiveness, ultimately contributing valuable insights to the broader discourse on advanced language models in educational technology.

4.1 BERT Score

In evaluating text generation for similarity matching, BERTScore emerges as a crucial metric. This assessment leverages the power of pre-trained contextual embeddings from BERT, employing them to match words in candidate and reference sentences based on cosine similarity. By comparing these nuanced meanings, BERTScore provides a robust measure of the coherence and semantic alignment between the generated and reference sentences, offering valuable insights into the quality of text generation.

In the evaluation process, the RAG agent is tasked with generating six responses corresponding to six distinct questions related to the lecture material, representing potential inquiries from students engaged in the course. For each question, a reference answer is manually provided that we believe accurately answers the question. The assessment involves comparing the generated answers with reference answers, and the BERT score is calculated as a metric to gauge the quality of the textual response. Across the BERT score results for each question, an average of 0.674, 0.600 and 0.635 is obtained for the precision, recall and f1-score respectively. This suggests that there is substantial text quality in the responses generated by the RAG agent.

4.2 Comparison with Baseline Models

The comparison with baseline models constitutes a critical aspect of our evaluation, providing a benchmark for understanding the relative performance of the generative AI model in the broader landscape. By contrasting our model against commonly used baseline models in similar applications, we gain insights into its strengths and areas for improvement. For the baseline models, the Zero-shot ReAct agent and ChatGPT/GPT 3.5 (without RAG) were used.

The Zero-shot ReAct agent is a commonly used language generation model that can create realistic contexts even without being trained on specific data. We can compare the responses from both the Zero-shot ReAct agent and our RAG agent to evaluate its effectiveness in carrying out its duties as a tutor. In the comparative analysis, it is observed that the Zero-shot agent delivers succinct and straightforward responses, while the RAG agent offers more elaborate answers, including step-by-step explanations to user queries. This disparity suggests that the RAG agent exhibits superior performance in fulfilling its role as a tutor when compared to the Zero-shot agent. The capacity of the RAG agent to provide detailed explanations enhances its effectiveness in assisting users, contributing to a more comprehensive and informative interaction.

ChatGPT is known for its ability to generate coherent and contextually relevant responses in natural language, making it suitable for conversational AI applications. It can understand and generate human-like text across a wide range of topics and contexts. In the context of model comparison, it is evident that ChatGPT outperforms the Zero-shot agent by delivering more detailed responses. Despite this, the relevance of the responses to the lecture materials appears to be compromised. In contrast, the RAG agent stands out for its frequent references to the lecture materials when providing explanations. This distinction indicates that the RAG agent aligns more closely with the expectations of a tutor, exhibiting a higher level of performance compared to the standard ChatGPT/GPT-3.5 model. The observed patterns suggest that while ChatGPT excels in providing detailed information, the RAG agent excels in maintaining relevance to the specific lecture materials, a crucial factor in the context of educational applications.

4.3 User Feedback

The user feedback portion of our evaluation serves as a crucial avenue for obtaining valuable insights into the practical usability and user satisfaction with the generative AI model. Through user satisfaction surveys, student participants are invited to provide ratings on key aspects such as coherence, relevance, and overall satisfaction with the generated content. This qualitative feedback is important in capturing the nuanced perspectives of end-users and understanding their experiences with the model in real-world scenarios. By incorporating user feedback into the evaluation, we aim to ensure that the generative AI model not only meets technical benchmarks but also aligns with user expectations and preferences, ultimately enhancing its effectiveness and user acceptance.

In this evaluative study, 17 students actively participated by providing their respective course materials, which were subsequently stored as embeddings in the vector stores utilized by the RAG agent for processing. Following the training of the agent, students were granted the opportunity to assess its performance and effectiveness as a tutor in their individual courses. Each student assigned a rating on a scale of 1 to 5 for coherence, relevance, and overall satisfaction with the generated content, where 1 denoted the least satisfaction and 5 represented the highest. The results revealed an average of 4.65 for coherence, 4.47 for relevance, and 4.53 for overall satisfaction. Noteworthy observations from students included the model's impressive ability to reference lecture materials and its feature of generating pertinent questions to assess comprehension. In contrast, a comparative survey involving the Zero-shot agent and ChatGPT model indicated a lower performance perception by students, with both models receiving lower scores in comparison to the RAG agent.

5. DISCUSSION

The comprehensive evaluation of the generative AI model reveals a highly satisfactory performance in its role as an educational tutor, excelling in the domain of deep learning and also across diverse courses. The RAG agent exhibits commendable proficiency by delivering detailed explanations and incorporating relevant references to lecture materials in response to student queries. Furthermore, the agent demonstrates its ability to assess student comprehension by posing pertinent questions, drawing on references to the initial question, the generated response, and associated lecture materials. The model's substantial BERT score substantiates its high text quality, aligning with expectations for informative and contextually relevant content. Comparative analysis against alternative models, including the Zero-shot ReAct agent and the ChatGPT/GPT 3.5 model, consistently indicate the superior performance of the RAG agent in fulfilling its tutor duties. This observation is reinforced by user feedback, with users expressing heightened satisfaction with the RAG agent compared to other models, further affirming its positive impact on the overall learning experience.

5.1 Limitations and Assumptions

In the training of the RAG agent, certain assumptions underpin the model's performance. It is assumed that the documents retrieved during training are relevant to the generation task; however, as the content of a course evolves, existing lecture materials may become outdated. Consequently, the RAG agent necessitates retraining with updated course materials to ensure continued effectiveness, given its reliance on retrieving contextually relevant information. Additionally, an assumption is made regarding the capability of pre-trained embeddings to convey meaningful representations from the course materials, thereby facilitating improved contextual understanding. The model also assumes that the utilization of prompt templates, such as those in Chain of Thought (CoT) reasoning, enhances generation performance by guiding the model to structure coherent and contextually relevant responses. Nevertheless, in practical scenarios, tutors may adopt varied methods to guide students on a case-by-case basis, challenging the agent's constant adherence to a prompt template, even though it may be effective in a general setting.

5.2 Applicability and Scalability

In the context of its tutoring duties and with a clear understanding of its role and relevant materials, the agent demonstrates effective performance. It is important to acknowledge the diversity in students' learning styles and preferences, factors that can influence their expectations from a tutor. For instance, certain learners may thrive with a more supported approach involving direct instruction, preferring in-depth explanations with fewer evaluative questions to assess comprehension. Conversely, other learners may lean towards a more independent and self-directed learning style, opting for concise explanations accompanied by a greater number of evaluative questions. Implementing these adjustments to cater to individual preferences is feasible but requires personalized fine-tuning of the model to align with each student's ideal response type.

The responses are currently presented in a textual format, a mode of delivery that may not be optimal for individuals with a preference for visual learning, where information is better absorbed through visualizations. To address this consideration in the future development of AI models, there exists the potential to integrate visual elements directly into the generated responses. This enhancement would serve to better accommodate the diverse learning styles of students, offering a more personalized and effective learning experience by catering to those who benefit from visual representations of information.

6. FUTURE WORK

For future iterations, this tutor agent can be combined with other intelligent agents such as professor agents or student agents to be integrated into a large-scale, comprehensive multi-agent conversational model. Through the reasoning model, users can gain valuable insights into the thought processes behind responses and foster a deeper understanding of the learning content via conversation with multiple agents in real-time. However, there exists limitations whereby errors can occur during the intermediate reasoning steps for complex tasks and lead to incorrect answers.

One suggestion to enhance the reasoning capabilities of the LLM to overcome this is through the use of step-back prompting, which introduces a level of abstraction. [4] This is done through encouraging the model to step back and derive high-level concepts and principles before reasoning through detailed instances to tackle the original question. This would prove especially useful in the realm of education whereby certain concepts would benefit from such an approach. Exploring ways to seamlessly integrate this into our existing model warrants additional research efforts.

7. CONCLUSION

Employing LLM as a tutor agent appears to be a well-received approach to tackling the issue of varied learning needs of individuals. However, the implications of such implementation should be further tested and evaluated to assess whether such a tutor agent actually aids in a student's learning journey and allows students to gain proficiency faster as compared to the conventional education method. This measurement would require prolonged experimentation on a large group of participants which is unavailable in the current situation. Furthermore, this conjecture is very similar to a decade-old debate over how the switch to e-dictionaries affects the long-term efficacy of language mastery.

In conclusion, in the short to medium term, an LLM-based tutor agent is expected to be popular among students due to factors such as convenience and feelings of self-complacency and assurance from its usage. However, further study would be required to determine its efficacy using standardized measurements of its true impact.

8. GITHUB REPOSITORY

https://github.com/yanrong19/Creating-Tutor-Agent-with-LangChain

9. REFERENCES

[1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, Denny Zhou. *Chain-of-thought prompting elicits reasoning in large language models*. https://arxiv.org/abs/2201.11903

[2] Courtland Leer, Vincent Trost, Vineeth Voruganti. *Violation of Expectation via Metacognitive Prompting Reduces Theory of Mind Prediction Error in Large Language Models.* https://arxiv.org/pdf/2310.06983.pdf

[3] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Awadallah, Ryen W. White, Doug Burger, Chi Wang. *AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation.* https://arxiv.org/pdf/2308.08155.pdf

[4] Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, Denny Zhou. *TAKE A STEP BACK: EVOKING REASONING VIA ABSTRACTION IN LARGE LANGUAGE MODELS.* https://arxiv.org/pdf/2310.06117.pdf