

Barkplug 2.0 and Beyond - a Chatbot for Assisting Students in High DFW Courses

Dr. Jason M. Keith, Iowa State University of Science and Technology

Jason Keith is Senior Vice President and Provost at Iowa State University. Prior to this appointment, he was Dean of the Bagley College of Engineering at Mississippi State University. Keith was recognized in 2014 as a Fellow of ASEE.

Dr. Amin Amirlatifi, Mississippi State University

Sudip Mittal, Mississippi State University

Subash Neupane, Mississippi State University

HIMANSHU TRIPATHI, Mississippi State University

Barkplug 2.0 and Beyond: a Chatbot for Assisting Students in High DFW Courses

Abstract

Higher education continues to respond to the challenges and opportunities presented by artificial intelligence (AI) and large language models (LLM) such as ChatGPT. In our prior work we introduced a chatbot that used AI and LLM to recruit prospective students, assist current students with academic advising (course selection, changing majors) and student affairs (directing students to university resources regarding the campus community, housing and dining, student organizations, mental health and more). Towards the promotion of student success initiatives we report in this work our formulation of course specific teaching assistants for engineering and computer science. Through inquiry-aided interaction, our chatbot helps students assemble prompts that aid them in the most challenging topics within core courses, and allow them to understand methodologies for solving homework problems and prepare for midterm exams. This paper will specifically discuss algorithm development and outline some case studies relevant to improving persistence, retention and graduation. Furthermore, while LLMs are generally considered a good tool for inferences and explanations, they struggle in formulating solutions to prompts that require calculations and involved engineering applications. This work highlights steps that would allow LLMs to work as reliable teaching assistants, serving both science and engineering oriented students.

Introduction and Literature Review

Artificial Intelligence (AI) continues to be one of the most talked-about topics in higher education and is of particular emphasis in colleges of engineering and computing across the world. In addition to being highly relevant in research themes during the era of autonomy and big data, there are large benefits to education as well. Advantages to chatbots are numerous, but include being available 24/7 to students, being (potentially) anonymous to answer questions that may be considered bad or humiliating to the student who is asking, to being able to provide quizzes and other materials to test knowledge retention, and having a capacity to handle what may be considered to be too many questions for the professor or teaching assistant to be able to respond to in a timely manner.

One of the initial uses of AI for supporting teaching expanded the use of the Piazza Q&A platform by Georgia Tech [1] (named Jill Watson) and then Stanford University [2]. In this latter work an AI tool was trained to address student inquiries in a core computer science course using 1500 questions and answers archived from Piazza. The questions were categorized as being related to a course policy, related to homework or some other assignment, or about a conceptual question. The results showed that the bot did very well at answering policy questions, fared relatively well on assignment questions, but struggled with addressing conceptual queries, even though a digitized version of the textbook was used to help train the bot on fundamentals.

In his master's degree thesis, Saura [3] worked on developing an AI chatbot built within the R programming language. This work cited immediate access as the main driving force for creating this bot, and thus focused on being able to understand the question being asked (this was quite

anticipatory towards the now maturing field of prompt engineering) and develop a front-end user-friendly application as well as develop mechanisms to find / formulate a correct answer. This work recognized the impact that interactive AI assistants like Siri, Alexa and others had and also pointed towards Georgia Tech and Deakin University [4] as inspiring examples. The algorithm utilized neural networks to find pathways to the closest related answers in its database.

Gonda and Chu [5] outlined the ability of a conversational chatbot to help train teaching assistants during orientation sessions at The Hong Kong University of Science and Technology. Their goal was to provide personalized experiences for the students and allow for scalable support, improve efficiency and consistency across multiple sections of the same course. This was well received by the faculty, who felt more confident and comfortable in ensuring strong delivery despite teaching the same materials and answering similar questions, even when training graduate assistants in eighteen departments.

Hamam [6] provided a literature review of chatbot's use in higher education, with an emphasis on improving teaching and learning. Advantages cited include the bots ability to engage in a conversation with students, and respond to frequently asked questions with consistency in an inexpensive and efficient manner. Sandu and Gide found from student surveys that the students preferred interacting with the chatbot as opposed to other communication methods [7]. Haristiani and Danuwijaya [8] evaluated the use of chatbots to facilitate the learning of foreign languages. Similarly, Deng and Yu [9] found a significant impact on overall student learning outcomes and knowledge retention. Thus, it appears that chatbots should be further developed, tested and evaluated to support student learning.

Specific to computer science and related learning applications, Ma et al. [10] developed a bot for software engineering which allowed for coverage of additional course material while providing additional teaching assistants (both artificial and in person) support. The human TAs were able to better support group projects since the bots were able to effectively manage lower-level questions in a consistent, rapid manner. However, Schroeder [11] points to issues with limitations of the bots in answering only common questions. Alsafari [12] gives an outline of the advent of LLMs as powerful tools for higher education including student services and as teaching assistants. They argue that these tools are best poised at this time to handle complex student queries, which is the focus of our current efforts.

Methodology for Design and Development of the Chatbot

At Mississippi State University, efforts have been made in support of the institutional strategic plan to improve student success. In a prior set of papers [13] and [14], the authors developed a chatbot for the Bagley College of Engineering to recruit prospective students into the college. Furthermore, the chatbot algorithm was written in a way to ask a series of questions to undecided students and (behind the scenes) build a prompt to run through our database and give advice to the prospective student regarding one (or more) majors that they should consider. A final aspect of the chatbot is to direct students to resources in the Division of Student Affairs. We initially chose these facets for our algorithm because when students are enrolled in a major that feels right

for them, and that when feeling welcome via a support network, students are retained (and graduate) at higher rates.

The next step in our work is to now focus on improving student success inside of classes that have particularly high DFW rates. Unique to our authorship to this paper, we focused our efforts on “PTE 3953 Petroleum Reservoir Rock Properties and Fluid Flow” and “CSE 1011 Introduction to Computer Science & Engineering.” Students tend to struggle with some of the foundational concepts associated with these courses and thus leave these majors into another department in the college or across the university. Areas that students struggle with include basic understanding of applying engineering fundamentals to the nature of oil and gas reservoirs, exploration and drilling, such as applying Darcy’s law to understand fluid flow (e.g., oil, water or gas) through porous media underground. Furthermore, basic coding skills and understanding computer logic, loops, and conditional statements are a common issue. The rest of this paper will outline processes used and algorithmic development to help students navigate these challenges. In this paper, we will focus on a fundamental question in petroleum engineering and computer science courses to provide an illustrative example. fs

To address these challenges, we developed a Teaching Assistant (TA) chatbot (BarkPlug 2.0) that provides real-time support tailored to the needs of these students. The chatbot is designed to answer both theoretical questions, such as applying engineering principles, and numerical or code-based queries, such as solving problems involving coding logic or debugging scripts. The architecture as depicted in Figure.1 is modular, consisting of distinct yet interlinked components that classify and process queries effectively. These include a Query Router, a Retriever, a Concept Agent, a Code Generator, an Execution Agent, and a Feedback Mechanism, all orchestrated to interact seamlessly with LLMs for final response generation.

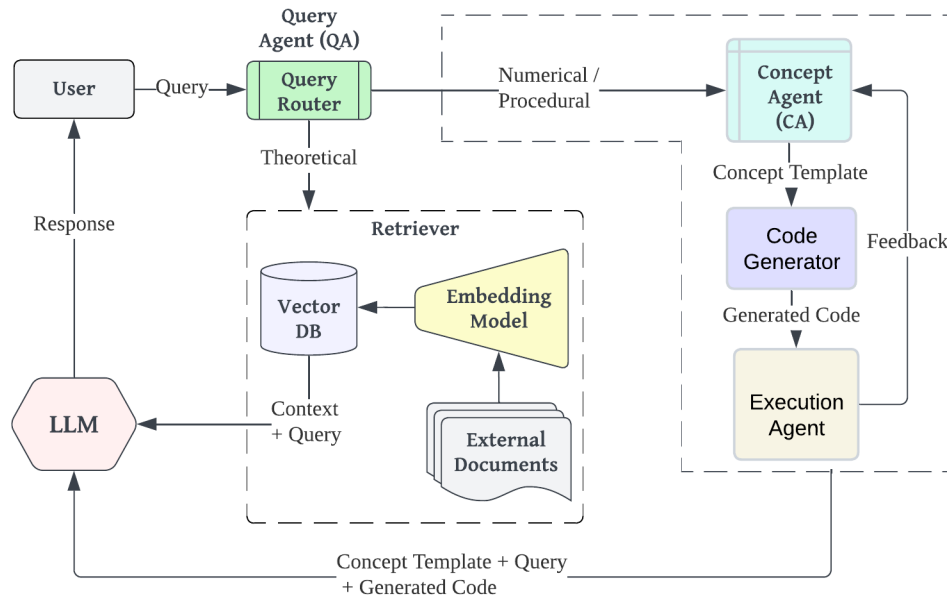


Figure 1: TA Chatbot Architecture (BarkPlug 2.0)

The first module is the Query Router, which acts as the entry point for the system. The objective of this module is to classify the user's query into one of two categories: theoretical or numerical/code based. It employs a lightweight classifier, such as a smaller fine-tuned LLM, to predict the query type based on its structure and keywords. The LLM is instructed to classify the query in a zero-shot setting. An example of a zero-shot prompt for query classification is illustrated in Figure 2.

```
prompt = f"""
    Instruction: You are an AI assistant trained to classify user queries into two categories: Theoretical or Numerical-Procedural. Follow these guidelines:

    1. Theoretical: The query involves abstract concepts, principles, or qualitative reasoning. It seeks explanations, definitions, or discussions without requiring calculations, numerical results, or step-by-step procedures.

    2. Numerical-Procedural: The query involves numbers, calculations, data analysis, code, or step-by-step instructions. It seeks specific numerical answers, solutions to equations, code implementations, or procedural guidance.

    Task: Read the user query below and classify it as either Theoretical or Numerical-Procedural. Provide only the classification as your output.

    User Query: {query}

    Output:
    """
```

Figure 2: Zero-shot prompt for query classification

When the query is classified as theoretical, the system employs a Retrieval-Augmented Generation (RAG) workflow to generate the response. First, it retrieves relevant information from a knowledge base (external knowledge sources), such as course curricula, textbooks, or scientific literature, focusing on key concepts related to the query. Next, a generative language model synthesizes the retrieved data into a coherent and detailed explanation, ensuring the response is accurate, contextually relevant, and tailored to the query. For example, as depicted in Table 1, for a user question like “*Concept Behind Initial Fluid Saturation in Reservoir*”, the RAG system begins by retrieving the top-k most relevant documents from a knowledge base, such as a Reservoir Rock Properties course or related textbooks. These documents, along with the user query, are then passed to an LLM for response generation. The LLM integrates the retrieved information to produce a comprehensive and scientifically accurate answer.

Table 1: Response generation to theoretical question using RAG workflow

Query	Query Classification	Response
Concept Behind Initial Fluid Saturation in Reservoir	Theoretical	The concept behind Initial Fluid Saturations in a reservoir involves the idea that pores are initially saturated with water ($S_w=1$), and hydrocarbons migrate into traps due to having a density less than water density. The distribution of hydrocarbons in the reservoir is such that gravity and capillary forces are in equilibrium. The minimum interstitial water saturation remains in the hydrocarbon zone even after accumulation occurs. Wettability plays a role, with water-wet conditions in drainage accumulation processes and oil-wet conditions in imbibition accumulation processes. Residual non-wetting phase saturation is also a factor in this concept.

For numerical or code-based problems, the system takes a more structured approach to ensure correctness and precision. Once the query is classified as a numerical or a code-related problem, it is routed to the Concept Agent (CA). The CA performs reasoning and generates a step-by-step procedure to solve the problem. For numerical queries, it may also include the final answer. The step-by-step reasoning can involve mathematical models, such as Darcy's law or fluid flow equations. For programming tasks, it may include algorithms for basic Python constructs, such as loops, conditionals, and common algorithms. The output of this module is a concept template. An example of concept template for a user query is illustrated in Figures 3 and 4.

```
user_query = " During a fluid saturations determination study, oil and water volumes were extracted from a core sample with bulk volume of 38.4 cm3, and porosity of 0.281. After applying necessary corrections, 5.12 cm3 of oil and 2.5 cm3 of water were recovered. Determine the fluid saturations."
```

Figure 3: Numerical question posed by user

To determine the fluid saturations, we first need to calculate the bulk volume occupied by oil and water and then calculate the fluid saturations. Here is the step-by-step algorithm to solve the problem:

1. Calculate the total bulk volume of fluids recovered:
 $\text{Total recovered volume} = \text{Volume of oil} + \text{Volume of water}$
 $\text{Total recovered volume} = 5.12 \text{ cm}^3 + 2.5 \text{ cm}^3$
 $\text{Total recovered volume} = 7.62 \text{ cm}^3$
2. Calculate the effective pore volume:
 $\text{Effective pore volume} = \text{Bulk volume of the core sample} \times \text{porosity}$
 $\text{Effective pore volume} = 38.4 \text{ cm}^3 \times 0.281$
 $\text{Effective pore volume} = 10.78 \text{ cm}^3$
3. Calculate the fluid saturations:
 $\text{Oil saturation} = (\text{Volume of oil} / \text{Effective pore volume}) \times 100\%$
 $\text{Oil saturation} = (5.12 \text{ cm}^3 / 10.78 \text{ cm}^3) \times 100\%$
 $\text{Oil saturation} = 0.475 \times 100\%$
 $\text{Oil saturation} = 47.5\%$
 $\text{Water saturation} = (\text{Volume of water} / \text{Effective pore volume}) \times 100\%$
 $\text{Water saturation} = (2.5 \text{ cm}^3 / 10.78 \text{ cm}^3) \times 100\%$
 $\text{Water saturation} = 0.232 \times 100\%$
 $\text{Water saturation} = 23.2\%$
4. Verify the calculations:
 $\text{Oil saturation} + \text{Water saturation} = 47.5\% + 23.2\% = 70.7\%$

As oil and water are the only two fluids present, the total fluid saturation should add up to 100%.

Therefore, the fluid saturations are: -

Oil saturation: 47.5%

Water saturation: 23.2%

Remaining pore space occupied by gas (Gas saturation): 29.3% (100% - 47.5% - 23.2%)

Figure 4. Concept template for numerical question illustrated in Figure 3

The input to the code generator is the concept template. It takes the concept template and generates a python code with basic python construct that can be executed without including complex dependencies. For example, in numerical problems, it applies relevant formulas provided in a concept template step by step. In coding tasks, it uses the series of logical constructs to create

python code. The output in either case is a Python code, which can solve the numerical or a coding problem. An example of generated code using for the same query and concept template from Figure. 3 and 4 are illustrated in Figure 5.

```

python
def calculate_fluid_saturation(volume_of_oil_recovered, volume_of_water_recovered, total_porosity):
    # Constants
    irreducible_water_saturation = 0

    # Step 1: Calculate the effective porosity
    effective_porosity = total_porosity - irreducible_water_saturation

    # Step 2: Calculate the volume of oil in the core sample
    oil_volume = volume_of_oil_recovered / effective_porosity

    # Step 3: Calculate the volume of water in the core sample
    water_volume = volume_of_water_recovered / effective_porosity

    # Step 4: Calculate the total fluid volume in the core sample
    total_fluid_volume = oil_volume + water_volume

    # Step 5: Calculate the oil saturation
    oil_saturation = oil_volume / total_fluid_volume

    # Step 6: Calculate the water saturation
    water_saturation = water_volume / total_fluid_volume

    return oil_saturation, water_saturation

# Test the function
oil_saturation, water_saturation = calculate_fluid_saturation(5.12, 2.5, 0.281)
print(f"The oil saturation in the core sample is approximately {oil_saturation * 100:.1f}%, and the water saturation is approximately {water_saturation * 100:.1f}%")

```

Figure 5: Generated code using concept template with step-by-step explanation

The input of the Execution Agent (EA) is the code generated by preceding agent. This agent then calculates the numerical outputs. The generated output code is then executed within a sandbox environment to verify its correctness. An example of code execution is illustrated in Figure 6.

```

oil_saturation_percent, water_saturation_percent, gas_saturation_percent = calculate_fluid_saturation(bulk_volume_cm3, porosity,
oil_volume_cm3, water_volume_cm3)

```

Figure 6: An example of execution agent in sand-box environment

If the output is incorrect (e.g., a runtime error in code or an invalid calculation), feedback is provided to the concept agent, which iteratively refines the concept template. An example of feedback with correct answer is depicted in Figure 7.

```

feedback = "the correct answers are {'Oil saturation': 47.4, 'Water saturation': 23.2, 'Remaining pore space (Gas saturation)': 29.4}"

```

Figure 7: An example of feedback with correct answer is provided as an input

Based on the feedback, the code is re-generated. An example of corrected code that accurately calculates oil, water, and gas saturation in percent is depicted as code snippet in Figure 8.

```
def calculate_fluid_saturation(bulk_volume_cm3, porosity, oil_volume_cm3, water_volume_cm3):
    total_pore_volume_cm3 = bulk_volume_cm3 * porosity
    oil_saturation = oil_volume_cm3 / total_pore_volume_cm3
    water_saturation = water_volume_cm3 / total_pore_volume_cm3
    gas_saturation = 1 - (oil_saturation + water_saturation)

    oil_saturation_percent = oil_saturation * 100
    water_saturation_percent = water_saturation * 100
    gas_saturation_percent = gas_saturation * 100

    return oil_saturation_percent, water_saturation_percent, gas_saturation_percent

bulk_volume_cm3 = 38.4
porosity = 0.281
oil_volume_cm3 = 5.12
water_volume_cm3 = 2.5

oil_saturation_percent, water_saturation_percent, gas_saturation_percent = calculate_fluid_saturation(
    bulk_volume_cm3, porosity, oil_volume_cm3, water_volume_cm3)

print(f"Oil saturation ≈ {oil_saturation_percent:.2f}%")
print(f"Water saturation ≈ {water_saturation_percent:.2f}%")
print(f"Gas saturation ≈ {gas_saturation_percent:.2f}%")

Oil saturation ≈ 47.45%
Water saturation ≈ 23.17%
Gas saturation ≈ 29.38%
```

Figure 8: Corrected code snippet after feedback

Once the code or numerical solution is validated through feedback mechanism then the generated code is combined with the query and the concept template and passed to the LLM. The LLM generates a final response, ensuring that the explanation is clear, accurate, and tailored to the user query. An example of an explanation for manual calculation of fluid saturations for oil and water is depicted in Figure 9.

One of the significant problems that students, and out of the box LLMs (such as ChatGPT, Gemini and CoPilot), face is dealing with engineering problems with mixed unit systems. Petroleum engineering, being a discipline where professionals have to work with units from multiple systems (Field, SI, Lab, etc.) in a single problem, is a prime example of this. In an attempt to evaluate the efficacy of different LLMs and quantify this issue, we performed an experiment where a petroleum engineering problem with mixed unit systems was fed into three different LLMs (ChatGPT, Gemini and CoPilot) and the result was compared with the ground truth (i.e., calculation by hand) as well as the output from our own model. We further informed LLMs that they have to pay attention to the units in the problem and observed significant improvements to their response (Table 2). However, despite several attempts in getting these engines to provide an error free response, we failed to achieve that with out of the box LLMs. The only instance in which we were able to get the correct answer, was after providing the correct answer to the engine, where it returned it back.

To manually calculate the fluid saturations for oil and water in a core sample based on the provided step-by-step algorithm, you can follow these steps:

1. Calculate the bulk volume of the core sample: Bulk Volume = 38.4 cm^3
2. Calculate the porosity of the core sample: Porosity = 0.281
3. Calculate the effective pore volume: Effective Pore Volume = Bulk Volume * Porosity = $38.4 * 0.281 = 10.79 \text{ cm}^3$
4. Determine the volume of water recovered: Volume of Water = 2.5 cm^3
5. Determine the volume of oil recovered: Volume of Oil = 5.12 cm^3
6. Calculate the water saturation: Water Saturation = Volume of Water / Effective Pore Volume = $2.5 / 10.79 = 0.2317$ or 23.17%
7. Calculate the oil saturation: Oil Saturation = Volume of Oil / Effective Pore Volume = $5.12 / 10.79 = 0.4745$ or 47.45%
8. Calculate the gas saturation (if applicable): Gas Saturation = $1 - (\text{Water Saturation} + \text{Oil Saturation}) = 1 - (0.2317 + 0.4745) = 0.2938$ or 29.38%

By following these steps, you can manually calculate the fluid saturations for oil and water in a core sample based on the provided algorithm.

Fig. 9: An example of step-by-step explanation of manual calculation of the fluid saturations for oil and water in a core sample.

To better illustrate and quantify this shortcoming in out of the box LLMs, Table 2 presents an instance of the iterative prompt refinement process applied to Gemini for solving the numerical flow rate calculation problem using Darcy's equation with mixed unit systems. Initially, Gemini provided an incorrect response of 4.9 barrels per day (bbl/day), resulting in a significant error of five orders of magnitude, indicating a fundamental misunderstanding of the unit system. Once we refined the prompt to instruct the LLM to use appropriate field units, Gemini improved its response in the second trial to 0.004900 bbl/day, reducing the error to ~11%. Further prompt refinements guided the LLM to apply Darcy's equation with the correct unit conversion factor, leading to an additional improvement in the third trial, where the error was reduced to ~10%. In the final iteration, after successive prompt adjustments and explicitly telling the model what the ground truth value was, Gemini reported back the correct response of 0.00552 in the required units of barrels per day.

Our TA chatbot, however, was trained to overcome this shortcoming and to pay close attention to the units, by converting everything to a consistent unit system before performing any calculations and converting the results back to the desired unit system once calculations are completed, thus providing the correct answer, which was equal to the ground truth, in the first trial for the same prompt that was provided to other models.

Table 2: An example of errors encountered by an out of the box LLM responses for flow rate calculation using Darcy's equation in a mixed units system setting

Question	Trail	Ground Truth	Response	Error %	Feedback (Remedy)
If you have 3 ft long core with 1 md permeability and you apply 14.7 psi pressure difference to the two ends, and the flowing fluid is water ($\mu = 1$ cP) and the core has a cross sectional area 1 ft ² , what would be the flow rate (bbl/day)?	1	0.00552	4.9	88668.12	-
	2	0.00552	0.004900	11.23	Asked to use field units
	3	0.00552	0.004965	10.05	Asked to use Darcy equation for field units which has the conversion factor of $1.127 * 10^{(-3)}$
	4	0.00552	0.00552	0.00	Gave it the correct answer of 0.00552!

Table 3 represents the response generation for programming related question, Figure 10 provides the actual code output to compute for nth number of Fibonacci sequence. Figure 11. illustrates pseudocode generation for the problem.

Table 3: Response generation to programming related questions

Query	Query Classification	Response
Write code for nth number of Fibonacci sequence code	Code	<p>Sure! Here is a step-by-step algorithm to find the nth number of the Fibonacci sequence:</p> <ol style="list-style-type: none"> 1. Define a function that takes an input <code>`n`</code> which represents the position of the Fibonacci number in the sequence. 2. Initialize two variables, <code>`a`</code> and <code>`b`</code>, to 0 and 1 respectively. These will represent the first two numbers in the sequence. 3. If <code>`n`</code> is 0, return 0. If <code>`n`</code> is 1, return 1. These are the base cases for the Fibonacci sequence. 4. Loop <code>`n-1`</code> times (since we have already considered the first two numbers in the sequence). <ol style="list-style-type: none"> a. Compute the next Fibonacci number by adding <code>`a`</code> and <code>`b`</code>. b. Update <code>`a`</code> and <code>`b`</code> to be the last two Fibonacci numbers calculated. 5. After the loop finishes, return the value of <code>`b`</code>, which is the nth Fibonacci number.

We will also briefly outline the implementation details, where we utilized two datasets for this research: the course curricula of “Petroleum Reservoir Rock Properties and Fluid Flow” and “Introduction to Computer Science and Engineering,” with the latter focused on python programming. All development was implemented using the LangChain framework, enabling seamless integration of data preprocessing, embedding, retrieval, and response generation. Course materials, provided as PDF files, were preprocessed using LangChain’s DirectoryLoader and PyPDFLoader. Documents were split into manageable chunks using the Recursive Character Text Splitter with a chunk size of 1000 characters and an overlap of 200 characters to preserve contextual integrity. These chunks were embedded using OpenAI’s *text-embedding-ada-002* model, and the resulting vectors were stored in Chroma DB, a high-performance vector database.

Here is a sample implementation of the algorithm in Python:

```
```python
def fibonacci(n):
 a, b = 0, 1

 if n == 0:
 return a
 if n == 1:
 return b

 for _ in range(n - 1):
 a, b = b, a + b

 return b

Example usage
n = 10
result = fibonacci(n)
print(f"The {n}th number in the Fibonacci sequence is: {result}")
```
```

This algorithm should efficiently compute the nth Fibonacci number. Let me know if you need further clarification or assistance!

Here is the Python function based on the provided algorithm to calculate the nth number of the Fibonacci sequence without any pre- or post-statements:

```
```python
def fibonacci(n):
 a, b = 0, 1

 if n == 0:
 return a
 if n == 1:
 return b

 for _ in range(n - 1):
 a, b = b, a + b

 return b
...`
```

Figure 10: Response generation to compute  $n^{\text{th}}$  Fibonacci number

You can use this `fibonacci` function to find the nth number in the Fibonacci sequence. Just provide the desired position `n` as an argument to the function.

Do you have a feedback? (yes or no): no

```plaintext

Pseudocode for Fibonacci Number Calculation Algorithm:

```
1. Procedure FibonacciNumber(n)
2.   Initialize variables a and b to 0 and 1 (representing the first two Fibonacci numbers)
3.   If n equals 0, return a
4.   If n equals 1, return b
5.   For i from 2 to n
6.     let temp = b
7.     b = a + b
8.     a = temp
9.   End For
10.  Return b
11. End Procedure
```

Example Usage:

```
n = 10
result = FibonacciNumber(n)
Output "The nth number in the Fibonacci sequence is: result"
...`
```

Figure 11: Sample pseudo-output for a given Fibonacci problem

For retrieval, we employed LangChain's vector store abstraction with similarity search configured to retrieve contextually relevant documents. OpenAI's GPT-4 API, integrated via LangChain, was used to generate accurate, context-aware responses. This pipeline ensured efficient and robust support for user queries, providing precise and detailed explanations tailored to the educational datasets.

Conclusion and Future Work

Our chatbot demonstrates the potential to transform educational support by leveraging LLMs and modular AI-driven architectures. This approach is unique in its integration of domain-specific problem-solving capabilities, such as engineering calculations and conceptual clarifications, into

a single system. Unlike conventional teaching assistants, our chatbot extends its utility to both theoretical and numerical problem-solving, offering personalized, context-aware guidance for students in high-DFW courses.

Through its deployment in “Reservoir Rock Properties and Fluid Flow” and “Introduction to Computer Science & Engineering” courses, the chatbot has addressed challenges such as coding fundamentals, logic comprehension, and applied engineering principles. These implementations have yielded promising results in enhancing student engagement and performance.

Our future works will go in two complementary directions. We will test and work to refine the bot based upon the impact that it has on students, and how it can best improve student success in engineering and computer science courses.

In the future, we aim to expand our chatbot’s capabilities to additional high DFW courses, such as engineering mechanics/statics (with problems such as bridge / truss analysis), as well as other computer science courses, (including data structures and algorithm optimization techniques, as well as cybersecurity and digital forensics). Further iterations of the chatbot will focus on improving its ability to facilitate collaborative learning, assist with project-based assessments, and provide actionable feedback to students and instructors.

References

- [1] Maderer, J. "Artificial Intelligence Course Creates AI Teaching Assistant," <https://news.gatech.edu/news/2016/05/09/artificial-intelligence-course-creates-ai-teaching-assistant>, May 2016, accessed January 2025.
- [2] Chopra, S., Gianforte, R., and Sholar, J. "Meet Percy: The CS 221 Teaching Assistant Chatbot," *ACM Transactions on Graphics*, Vol. 1 (1), December 2016.
- [3] Lluna, A. P. "Creation and Development of an AI Teaching Assistant," Master's Thesis, Universitat Politècnica de Catalunya, 2017/2018.
- [4] Hubert. "6 Ways Artificial Intelligence and Chatbots Are Changing Education," *Chatbots Magazine*, May 2017.
- [5] Gonda, D. E. and Chu, B. "Chatbot as a learning resource? Creating conversational bots as a supplement for teaching assistant training course," 2019 IEEE International Conference in Engineering, Technology and Education.
- [6] Hamam, D. "The New Teacher Assistant: A Review of Chatbots Use in Higher Education," in: Stephanidis, C., Antona, M., Ntoa, S. (eds) *HCI International 2021 - Posters*. HCII 2021. *Communications in Computer and Information Science*, vol 1421. Springer, Cham. https://doi.org/10.1007/978-3-030-78645-8_8.
- [7] Sandu, N., Gide, E.: Adoption of AI-chatbots to enhance student learning experience in higher education in India. In: 2019 18th International Conference on Information Technology Based Higher Education and Training (ITHET), pp. 1–5. IEEE, September 2019.
- [8] Haristiani, N.U.R.I.A., Danuwijaya, A.A.: Gengobot: A chatbot-based grammar application on mobile instant messaging as language learning medium. *J. Eng. Sci. Technol.* 14(6), 3158–3173 (2019).
- [9] Deng, X., & Yu, Z. (2023). A meta-analysis and systematic review of the effect of chatbot technology use in sustainable education. *Sustainability*, 15(4), 2940.
- [10] S. -P. Ma, Y. -C. Liang, S. -K. Wang, Y. -W. Huang and W. -L. You, "TABot: A Teaching Assistant Chatbot for Software Engineering Courses," 2023 30th Asia-Pacific Software Engineering Conference (APSEC), Seoul, Korea, Republic of, 2023, pp. 627-631, doi: 10.1109/APSEC60848.2023.00084
- [11] N. L. Schroeder and O. O. Adesope, "A case for the use of pedagogical agents in online learning environments," *Journal of Teaching and Learning with Technology*, no. 2, pp. 43-47, 2013.
- [12] Bashaer Alsafari, Eric Atwell, Aisha Walker, Martin Callaghan, Towards effective teaching assistants: From intent-based chatbots to LLM-powered teaching assistants, *Natural Language Processing Journal*, Volume 8, 2024,

[13] Keith, J. M., & Amirlatifi, A., & Rahimi, S., & Neupane, S., & Mittal, S. (2024, June), *Bark Plug: The ChatGPT of the Bagley College of Engineering at Mississippi State University* Paper presented at 2024 ASEE Annual Conference & Exposition, Portland, Oregon. 10.18260/1-2—46635.

[14] Neupane, S., & Hossain, E., & Keith, J. & Tripathi, H., & Ghiasi, F. & Goliarz, N. A., & Amirlatifi, A., & Mittal, S., & Rahimi, S. From Questions to Insightful Answers: Building an Informed Chatbot for University Resources Paper presented at 2024 Frontiers of Education, Washington D.C.