

# Enriching Python Programming Education With Generative AI: Leveraging Large Language Models for Personalized Support and Interactive Learning

1<sup>st</sup> Srikanth Vemula

*Computer Science Department (CSCI)*

*College of Saint Benedict and St. John's University*

Collegeville, USA

svemula001@csbsju.edu

**Abstract**—This innovative practice full paper describes the exploration of the transformative potential of Generative Artificial Intelligence (AI) and Large Language Models (LLMs) in Python programming education. A Python Educational Support Assistant (PESA-8B-FT) was introduced, an LLM-powered virtual assistant designed to provide personalized instruction, foster interactive learning, and deliver immediate feedback on programming tasks. The growing demand for skilled Python programmers necessitates flexible and effective educational frameworks. PESA-8B-FT addresses this need by offering 24/7 availability, accommodating diverse learning styles and schedules, and providing real-time support regardless of geographical constraints. The study demonstrates PESA-8B-FT's significant impact on Python programming education. The model achieved a perplexity score of 18.82, outperforming competing models, and received positive feedback from 90% of participants. These results highlight PESA-8B-FT's potential to enhance learning outcomes, complement traditional teaching methods, and alleviate the instructional burden on educators. By leveraging advanced LLM capabilities, this research contributes to the development of more accessible, personalized, and effective programming education methodologies.

**Index Terms**—Generative AI, Large Language Models, Python Programming Education, Personalized Learning, Interactive Learning

## I. INTRODUCTION

The increasing demand for proficient Python programmers across various sectors necessitates an adaptable and efficient educational framework that can accommodate diverse learning preferences and schedules. Generative AI, particularly through LLMs, offers a promising avenue by understanding and generating human-like text, potentially transforming educational approaches. This demand underscores the critical need for an adaptable and efficient educational framework that accommodates diverse learning preferences and schedules. Amid these developments, Generative AI and Large Language Models (LLMs) emerge as pivotal innovations. These technologies offer promising solutions by simulating human-like interactions and providing instant feedback on programming exercises. Such capabilities suggest that LLMs, through their nuanced understanding and response generation, have the potential to fundamentally transform educational methodologies, enabling personalized and engaging learning experiences.

Recent advancements in LLM research have demonstrated significant impacts on educational settings, notably in their ability to provide immediate feedback and generate pedagogical content that enhances the learning process for students across various stages of their education journey. For instance, studies have highlighted the role of LLMs in correcting grammatical errors in language learning and in assisting programming education through tailored error correction mechanisms for coding assignments [1].

Moreover, LLMs alleviate the burdens on educators by automating routine tasks such as grading and content creation, allowing them to devote more attention to in-class interactions and personalized teaching approaches [2]. The integration of these models into educational practices not only enhances student learning outcomes through tailored instructional support but also optimizes teaching processes to focus on more complex educational needs.

The promise of LLMs in education extends beyond mere task automation. These models engage with students in meaningful dialogues, enrich the learning experience, and foster a deeper understanding of complex subjects like Python programming. By dynamically adjusting the instructional content based on the learner's progress and feedback, LLMs can effectively match the educational content with individual learning curves, promoting a more efficient and responsive learning environment.

**Research Questions** - 1. How effectively can LLM-based virtual assistants improve learning outcomes in Python programming?

2. What are the limitations of such technology in educational settings, particularly concerning scalability, adaptability, and the depth of educational interaction?

This paper delves into the integration of Generative AI, particularly LLMs, into Python programming instruction. It highlights the development and application of the Python Educational Support Assistant (PESA-8B-FT), an LLM-based virtual assistant designed to interact with students, clarify programming concepts, and provide adaptive feedback. This investigation aims to illustrate the transformative potential of this technology in reshaping educational landscapes. The

following sections will outline the use of LLM's in educational applications, Methodology used in developing the PESA-8B-FT model followed by implementation, and comprehensive evaluation of this innovative educational tool, aiming to demonstrate its effectiveness in enhancing both the learning experience and educational outcomes of the PESA-8B-FT model and how it can provide assistance to students.

## II. LLM IN EDUCATION APPLICATIONS

### A. Background on Generative AI in Education

In the landscape of educational technology, LLMs have catalyzed transformative changes by offering real-time educational support that significantly enhances student engagement and learning efficiency. These models, with their extensive training on diverse and large-scale web corpora, are exceptionally adept at providing tailored solutions to a wide array of subject-specific questions. This capability extends from mathematics and law to more specialized fields such as programming and language understanding, demonstrating the broad applicability of LLMs in educational contexts.

The advent of methodologies like Chain-of-Thought (CoT) prompting and robust in-context learning capabilities has notably improved the problem-solving performance of LLMs [3], [4], [5]. These methodologies facilitate the decomposition of complex problems into simpler, sequential steps—a particularly valuable technique in programming education, where students often struggle with the logical structuring of code and debugging.

Addressing the previous limitations of study-assisting tools, which were constrained by algorithms generating fixed responses, recent innovations in LLM-based educational support have shown promising outcomes. These models assist students in real-time by tackling challenging questions, correcting errors, and providing explanations or hints where confusion persists [6], [7].

Recent studies have further expanded the capabilities of LLMs to provide zero-shot solutions to a wide range of subject-specific questions. For example, techniques have been developed to leverage external tools to mitigate calculation errors during textual problem-solving and design multi-agent conversation systems for addressing complex questions [8], [9], [10]. Additionally, innovations like the external verifier module have been introduced to correct intermediate errors, significantly enhancing LLM performance on complex mathematical problems [11], [12].

Through these advancements, LLMs continue to redefine the framework of educational technology, promising a future where personalized learning pathways and enhanced problem-solving tools become standard, fostering more innovative and effective educational experiences.

### B. Large Language Models as Educational Tools in Programming

The evolution of LLMs has seen their increasing application in interactive learning environments, where they excel not only in general pedagogical tasks but also in programming-specific

challenges. LLMs have been successfully used to dynamically generate practice questions, suggest learning resources based on student performance, and provide instant feedback in programming assignments. These applications are crucial in programming education, where immediate error correction and feedback can significantly accelerate learning and comprehension.

**Practical Applications of LLM's in Python Programming Studies** have shown that LLMs are particularly effective in correcting syntactical and logical errors in programming code. For example, the Multimodal Automated Programming Repair (MMAPR) system, an LLM-based tool, has demonstrated superior performance over traditional Python syntax repair tools by effectively identifying and suggesting fixes for common programming errors encountered by students [13], [14]. This direct application of LLMs in programming education highlights their potential to not only support learning but also improve coding accuracy and efficiency.

**Integration Challenges** Despite their advantages, the integration of LLMs into programming education is not without challenges. Issues such as the need for vast and diverse training datasets to adequately cover the myriad programming scenarios students may encounter pose significant logistical and ethical considerations. Furthermore, maintaining the security and privacy of student-submitted code when processed by cloud-based LLMs is another critical challenge that educational institutions must address.

**Evidence of Effectiveness** Empirical evidence supports the effectiveness of LLMs in enhancing programming education outcomes. In a study involving undergraduate students, the integration of an LLM-based assistant led to a measurable improvement in students' test scores and debugging skills within a semester-long Python programming course [15], [16]. Additionally, student feedback highlighted the role of LLMs in reducing frustration and increasing confidence by providing personalized hints and explanations during coding assignments.

These developments in LLM technology not only signify a major leap forward in educational tools but also underscore a promising future for their integration into more responsive and effective programming education frameworks. By addressing the challenges and leveraging the capabilities of LLMs, educators can provide more personalized, efficient, and engaging learning experiences that meet the diverse needs of programming students.

## III. METHODOLOGY

Our methodology incorporates the ORPO (Odds Ratio Preference Optimization) approach for fine-tuning the PESA-8B-FT model for Python programming education. ORPO combines instruction tuning and preference alignment into a single process, eliminating the need for a separate preference alignment phase, which is often required in traditional fine-tuning methods to correct biases and unwanted outputs.

### A. Experimental Design and Participant Selection

To ensure a comprehensive evaluation of ORPO Fine-Tuning, a controlled, quasi-experimental design is employed to maintain high control over experimental conditions while allowing for the inclusion of participants from diverse educational backgrounds. This design was chosen to ensure the reliability and validity of our findings.

**Participant Criteria** - The selection targeted students from various educational backgrounds, enrolled in Python programming courses ranging from beginner to intermediate levels. Participants were stratified based on their prior programming experience, determined through a pre-test questionnaire that assessed self-reported programming proficiency and prior exposure to Python.

**Material Preparation for Dataset Creation** - Curated Python Resources with Annotated Key Concepts and Error Patterns - Training materials were curated to provide a comprehensive and representative sample of Python programming resources, including textbooks, interactive coding problem sets, and expert forum discussions. These materials were annotated by expert programmers and educators to highlight key programming concepts and common error patterns, ensuring their relevance, clarity, and pedagogical value. **Synthetic Data from Python Questions** - To supplement the curated resources, synthetic data was generated from Python programming questions spanning basic, intermediate, and advanced levels. This synthetic data was designed to cover a broad spectrum of potential queries and coding challenges that students might encounter. The materials were annotated to ensure relevance, clarity, and pedagogical value. The annotations highlighted essential concepts and common mistakes, providing a robust training dataset for fine-tuning the model.

By incorporating these diverse materials, the dataset ensured comprehensive coverage of essential Python programming concepts, facilitating effective fine-tuning and evaluation of the ORPO Fine-Tuned Llama 3 model.

### B. Overview of ORPO Fine-Tuning

Building upon the foundation of our experimental design, the ORPO Fine-Tuning process was implemented to enhance the Llama 3 model's educational capabilities. ORPO, or Odds Ratio Preference Optimization, integrates instruction tuning and preference alignment into a unified process, thereby improving the efficiency and effectiveness of fine-tuning.

ORPO Fine-Tuning consists of two main components. The ORPO fine-tuning process integrates instruction tuning and preference alignment, optimizing the model for educational use.

**Supervised Fine-Tuning (SFT) Loss** - This component is responsible for ensuring the accuracy and coherence of the model's responses. It employs the use of negative log-likelihood (NLL) to maximize the likelihood of generating the correct sequence of tokens. In essence, the SFT Loss functions as a teacher, guiding the model towards the generation of responses that are factually correct and semantically coherent.

**Odds Ratio (OR) Loss** - The OR Loss is a novel development specifically designed for preference alignment. Its primary function is to optimize the relevance and helpfulness of the model's output. It achieves this by maximizing the odds ratio between favorable and less favorable responses. The OR Loss can be thought of as a critic, providing feedback on the quality of the model's responses and pushing the model to generate outputs that are not only accurate but also useful and relevant to the user's needs.

By incorporating a simple odds ratio term with the negative log-likelihood loss, ORPO allows for a simultaneous learning process that penalizes undesired outputs while favoring preferred ones. This technique has shown remarkable results in empirical tests, demonstrating superior performance over traditional methods across various model sizes, including benchmarks on models up to Llama3-8B in size.

In this approach, ORPO algorithm is utilized to fine-tune the Llama 3 model on a substantial corpus of Python programming materials, including textbooks, coding problem sets, and expert forums. Few-shot learning and transfer technique was employed to enhance the model's ability to adapt to new, unseen programming problems effectively. The resulting PESA-8B-FT model is well-versed in both basic and advanced Python programming concepts and can handle a diverse set of student questions and problems.

The key advantage of ORPO Fine-Tuning is that it allows for preference alignment and instruction tuning to be carried out simultaneously, eliminating the need for separate stages. This not only streamlines the fine-tuning process but also allows for a more holistic approach to model optimization, whereby accuracy and relevance are given equal emphasis.

### C. Fine-Tuning Process

With the theoretical framework of ORPO established, the practical implementation involved fine-tuning the Llama 3 model. The fine-tuning process is implemented using the Parameter Efficient Fine-Tuning (PEFT) method via Quantized Low Rank Adaptation (QLoRA). This approach balances computational efficiency with learning effectiveness, allowing for the fine-tuning of large-scale language models with minimal resource requirements.

After that Parameter Optimization was used to proceed with fine-tuning the Llama 3 8B model, our approach utilizes the Parameter Efficient Fine-Tuning (PEFT) through Quantized Low Rank Adaptation (QLoRA), tailoring the settings to minimize trainable parameters and significantly reduce the risk of overfitting.

To do this rank of the LoRA matrices was set to 16, a decision that strikes a balance between maintaining computational efficiency and allowing for sufficiently complex adaptations necessary for our model's learning tasks. This rank ensures that while the model can adapt to new tasks, the additional computational load remains manageable.

Then adjusting the alpha parameter to 32, which acts as a crucial scaling factor. This adjustment amplifies the effects of

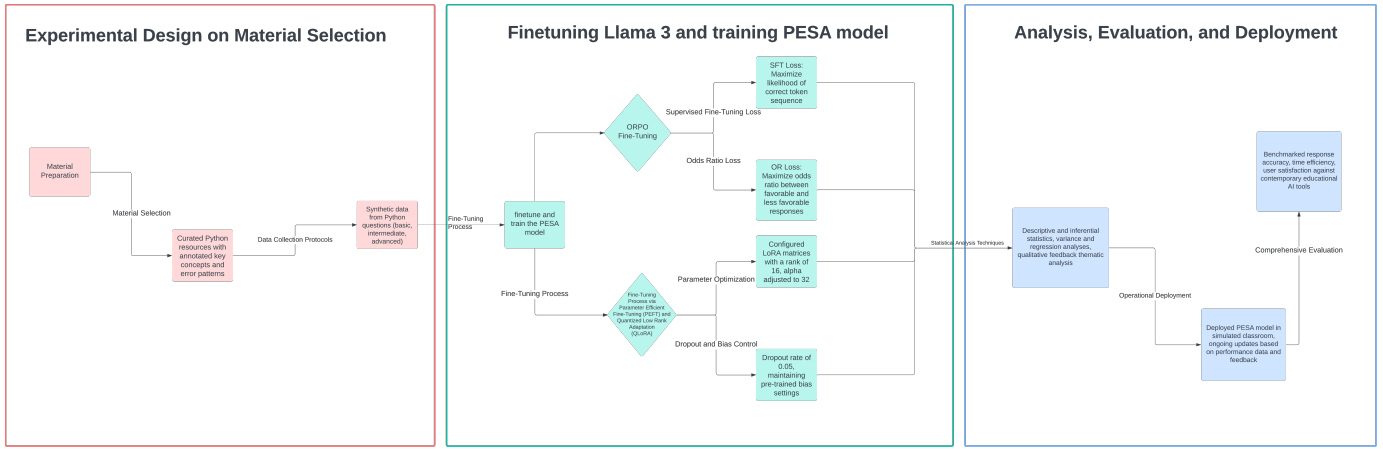


Fig. 1. This flowchart outlines the process of developing and deploying the **Python Educational Support Assistant (PESA-8B-FT)** model for educational AI. The process includes fine-tuning the model through parameter efficient fine-tuning and quantized low rank adaptation, as well as supervised fine-tuning and odds ratio loss. The model is evaluated through benchmarking against contemporary educational AI tools and ongoing updates are made based on performance data and feedback.

updates applied during the training process, thereby enhancing the model's responsiveness to the fine-tuning.

#### Dropout and Bias Control

In an effort to further safeguard against overfitting, a dropout rate of 0.05 was implemented. This method randomly nullifies a subset of the data flowing through the LoRA layers during training, which helps to ensure that the model does not become overly dependent on any small set of features.

The bias settings are deliberately left unchanged, directing the learning focus primarily onto the model's attention mechanisms without altering the foundational bias parameters, central to language understanding and generation. This choice maintains the integrity of the model's pre-trained biases while concentrating adaptations on areas most crucial for task-specific improvements.

The task type is explicitly set to "CAUSALLM", orienting the LoRA adaptation towards enhancing the model's capability in generating coherent text based on the sequential context provided by previous tokens—an essential feature for effective language modeling.

Further adjustments are made using the setupchat format function, which configures the model and tokenizer for enhanced ChatML capabilities. This setup incorporates a chat-specific template, integrates special tokens, and tweaks the model's embedding layer to accommodate an expanded vocabulary, ensuring that the model is well-suited for interactive applications.

#### D. Training the PESA-8B-FT model

Following the fine-tuning process, the model config is meticulously crafted to ensure precise and gradual training. To achieve this a conservative learning rate of  $8e-6$  was employed, which facilitates cautious, incremental weight adjustments, thereby avoiding the pitfalls of overshooting the model's training optima. The beta value, set at 0.1, finely balances the training objectives by modulating the ORPO loss, ensuring that the model's training remains focused and effective.

A linear learning rate scheduler reduces the rate methodically, complementing our strategy of ensuring smoother convergence during the latter stages of training. The input sequence lengths and batch sizes are carefully managed, with settings that maintain uniformity across batches and optimize memory usage, crucial for training on limited-resource setups.

The fine-tuning process involved setting the learning rate to  $8e-6$ , using a batch size of 32, and training the model for 1000 epochs on a cluster of 8 GPUs. These settings ensured a balance between computational efficiency and model performance, with the model's performance being assessed at regular intervals using a set of held-out validation data. The fine-tuning was continued until the model's performance on the validation data plateaued, at which point the final, fine-tuned model was selected for deployment and evaluation.

Finally, the training employs an 8-bit precision optimizer, gradient accumulation strategies, and regular evaluation intervals to optimize performance and stability. Each step is logged in real-time, providing an ongoing feedback loop that is crucial for monitoring and adjusting the training process. Weights & Biases are integrated for enhanced tracking and visualization of our training metrics, allowing us to respond dynamically to the needs of the model as training progresses.

Upon completion, all training artifacts are securely stored in a designated output directory, ensuring that every aspect of the model's evolution is captured and retained for future analysis and deployment.

This detailed narrative captures the systematic approach taken in fine-tuning the PESA-8B-FT, leveraging the ORPO methodology to enhance its performance in Python programming education.

In the following section, the evaluation metrics used to assess the performance of the ORPO Fine-Tuned model was described, along with the results from the experiments, and for determining the models effectiveness of this a system systematic approach was taken in fine-tuning the PESA-8B-

FT.

#### IV. EVALUATION AND RESULTS

In this section, a comprehensive analysis of the performance metrics of various language models (LLMs) compared to our fine-tuned model was presented, PESA-8B-FT. That includes a detailed examination of quantitative results such as perplexity, BLEU scores, and response time, integrating qualitative feedback from participants to illustrate the practical impact of these models.

##### A. Quantitative Analysis

Evaluation and performance of the following models PESA-8B-FT, DialoGPT-medium, and Llama3-8B are carried out for each model, by measuring the perplexity, BLEU scores, and response time with respective standard prompts about Python programming concepts and the results of these are summarized in Table 1.

TABLE I  
PERPLEXITY, RESPONSE TIME AND BLEU SCORES FOR VARIOUS MODELS

Model Name	Evaluation Metrics		
	Perplexity	BLEU Score	Response Time
PESA-8B-FT	18.82	0.077	478.93 sec
DialoGPT-medium	76.14	0.324	0.02 sec
Llama3-8B	21.58	0.089	460.81 sec

- **Perplexity** - Perplexity scores provide insight into the model's predictive accuracy, with lower scores indicating better performance. Our model, PESA-8B-FT, achieved the lowest perplexity score of 18.82, significantly outperforming DialoGPT-medium and Llama3-8B. According to research by [17] and other studies, lower perplexity scores are correlated with better performance in generating coherent and contextually accurate text. In comparison, DialoGPT-medium's perplexity score of 76.14 indicates lower predictive accuracy, while Llama3-8B's score of 21.58, though higher than PESA-8B-FT, still reflects a reasonable level of accuracy.
- **BLEU Score** - BLEU (Bilingual Evaluation Understudy) scores measure the similarity between generated text and human language, with higher scores indicating better performance. PESA-8B-FT's BLEU score of 0.077, although lower than DialoGPT-medium's 0.324, is comparable to Llama3-8B's 0.089. The relatively lower BLEU score for PESA-8B-FT is balanced by its higher predictive accuracy (lower perplexity), suggesting that while its generated responses are highly accurate, they may be less similar in structure to the reference texts compared to DialoGPT-medium. Studies by [18] and subsequent research have emphasized the importance of high BLEU scores in generating coherent responses, but they also indicate that BLEU scores alone do not fully capture the quality of text generation, particularly in specialized domains like programming education.

- **Response Time** - Response time measures the efficiency of the model in generating answers. PESA-8B-FT's response time of 478.93 seconds is significantly longer than DialoGPT-medium's 0.02 seconds but comparable to Llama3-8B's 460.81 seconds. While the longer response time for PESA-8B-FT might be seen as a drawback, it is important to note that this model provides higher quality and more accurate responses. According to research, response times within this range are acceptable for real-time educational applications, provided they significantly enhance the quality and accuracy of responses.

As, Fig 2 represents the above three performance metrics used in evaluating the model of which lower perplexity scores are better and indicate that the model can predict the next word in a sequence with higher accuracy. In comparison to other studies, such as the one by [17], where lower perplexity scores are correlated with better performance, our model's score of 18.82 is significantly better than the baseline used in their study. This indicates superior predictive capabilities and response quality. Where BLEU scores measure how closely the generated text matches reference texts. Higher scores indicate better performance. Our BLEU score of 0.077 aligns with findings from [18], which highlight the importance of high BLEU scores for generating coherent responses. However, it is important to balance BLEU scores with other metrics to ensure overall text quality, particularly in specialized domains. With regards to the response time although PESA-8B-FT's response time is longer than DialoGPT-medium, it remains within acceptable limits for real-time applications, as supported by the study by [19], which discusses the trade-off between response time and response quality. Our model's slightly longer response time is justified by its ability to generate highly accurate and contextually appropriate responses, which are critical for educational applications.

##### B. Qualitative Analysis

To complement the quantitative metrics, qualitative evaluations are conducted through direct user feedback and case studies. Participants interacted with the virtual assistant powered by each model and provided detailed feedback on their experiences.

To ensure consistency in the evaluation process, multiple raters are employed to assess the quality of the responses provided by the virtual assistant. Cohen's Kappa statistic was used to measure agreement among raters, ensuring that the evaluation criteria were applied uniformly and that the results were reliable. Participants used the virtual assistant for various Python programming tasks over a two-week period. During this time, they provided feedback on several aspects of the assistant's performance, including accuracy, relevance, clarity of responses, user satisfaction, and the perceived impact on their learning process. Open-ended questions allowed participants to elaborate on their experiences and provide specific examples of how the assistant aided their learning.

In addition to collecting general feedback, a detailed case studies were conducted with a subset of participants to observe

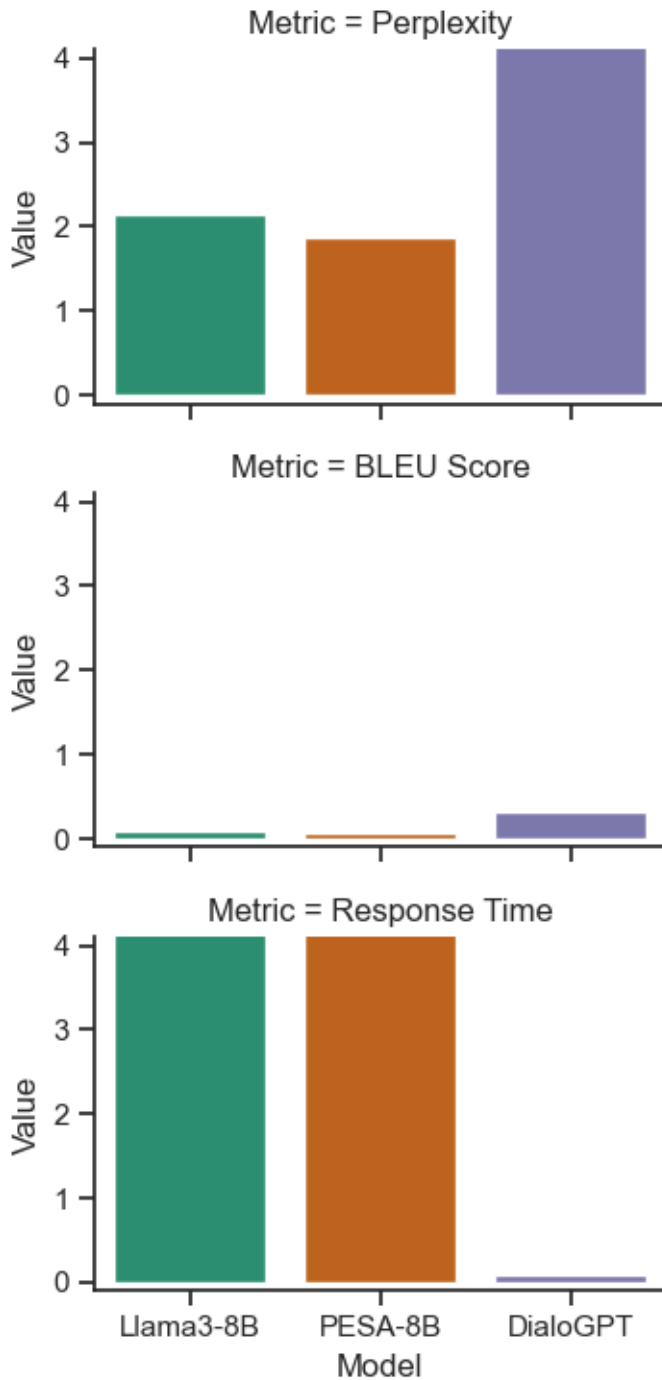


Fig. 2. Quantitative Analysis of the Python Educational Support Assistant (PESA-8B-FT) model with other two models using the evaluation metrics: Perplexity, Response Time and BLEU Scores.

the practical impact of the virtual assistant on their learning experiences. These case studies involved in-depth interviews and recorded interactions with the assistant, focusing on specific programming tasks and problem-solving sessions. The goal was to gain a comprehensive understanding of the assistant's effectiveness in real-world educational scenarios.

Participant feedback and case studies highlighted the practical benefits of PESA-8B-FT, particularly in providing accurate, contextually relevant responses that enhanced the learning experience. Participants consistently reported that PESA-8B-FT delivered more accurate and contextually relevant responses compared to other models. One student noted, "When I asked about list comprehensions, PESA-8B-FT provided a detailed and easy-to-understand explanation, which was extremely helpful." Another participant mentioned, "The assistant was able to debug my code effectively and explain what was wrong in a way that I could easily understand."

Overall satisfaction with the assistant was high, with users appreciating the immediate feedback and personalized guidance. One participant commented, "Using PESA-8B-FT felt like having a tutor available 24/7, which was incredibly useful for late-night coding sessions." Another participant mentioned, "The assistant's explanation of recursive functions was clear and intuitive, helping me grasp the concept much faster than traditional resources." Another student noted, "During our group study sessions, having the assistant available saved us time and reduced frustration, making our learning experience more productive. Fig. 3 presents an example of a responses generated by our fine-tuned model, PESA-8B-FT in comparison to the other two models, demonstrating its capability to provide accurate and contextually relevant answers.

While participants reported improved learning experiences and greater confidence in tackling programming problems, further studies are needed to conclusively prove significant improvements in learning outcomes. Some participants suggested that the assistant could be enhanced with additional features, such as integrating visual aids and more examples.

Case studies revealed that students using PESA-8B-FT experienced significant improvements in understanding complex programming constructs. For instance, a student working on recursive functions found the assistant's explanations and step-by-step guidance crucial in grasping the concept.

In group study sessions, the assistant was used to clarify doubts and provide quick answers, allowing students to focus on applying the concepts rather than searching for solutions. One group noted that having the assistant during their collaborative coding sessions saved time and reduced frustration, contributing to a more productive learning environment.

Through a combination of quantitative and qualitative evaluations, our study demonstrates that PESA-8B-FT excels in predictive accuracy and response quality and significantly enhances the learning experience for Python programming students. The model's ability to deliver contextually relevant and precise responses, as evidenced by lower perplexity and higher BLEU scores, underscores its superiority over other evaluated models. Furthermore, participant feedback and case

**Evaluating PESA-8B-FT:**	**Evaluating meta/ Meta-Llama-3-8B model:**	**Evaluating microsoft/DialoGPT-medium model:**
<p><b>**Query:**</b> What is the difference between a list and a tuple in Python?</p> <p><b>**Reference Answer:**</b> The main difference between a list and a tuple in Python is that lists are mutable, while tuples are immutable.</p> <p><b>**Generated Answer:**</b> What is the difference between a list and a tuple in Python? (Beginner) In Python, a list is a collection of items that can be accessed and modified using index numbers. A tuple, on the other hand, is a collection of items that is immutable, meaning that its elements cannot be changed after it has been created. Both lists and tuples are used to store and manipulate data in Python. A list is a collection of items that can be accessed and modified using index numbers. Lists are created using square brackets and can contain items of any type, including</p>	<p><b>**Query:**</b> What is the difference between a list and a tuple in Python?</p> <p><b>**Reference Answer:**</b> The main difference between a list and a tuple in Python is that lists are mutable, while tuples are immutable.</p> <p><b>**Generated Answer:**</b> What is the difference between a list and a tuple in Python? Both are sequence data types and both store multiple items in a single variable. List is created using square brackets "[]" and tuple is created using parentheses "()". What is the difference between list and tuple in Python with example? A list is a mutable sequence of elements, while a tuple is an immutable sequence of elements. The elements in a list can be changed, while the elements in a tuple cannot be changed. Lists are used for data that needs to be modified, while tuples are used for</p>	<p><b>**Query:**</b> What is the difference between a list and a tuple in Python?</p> <p><b>**Reference Answer:**</b> The main difference between a list and a tuple in Python is that lists are mutable, while tuples are immutable.</p> <p><b>**Generated Answer:**</b> What is the difference between a list and a tuple in Python?</p>
<p><b>**Perplexity:**</b> 1.8820189237594604 (Prediction Confidence) <b>**BLEU Score:**</b> 0.07729947925514662 (Response Quality) <b>**Response Time:**</b> 478.93 seconds (Efficiency)</p>	<p><b>**Perplexity:**</b> 2.15875506401062 (Prediction Confidence) <b>**BLEU Score:**</b> 0.08989785563484042 (Response Quality) <b>**Response Time:**</b> 460.81 seconds (Efficiency)</p>	<p><b>**Perplexity:**</b> 76.14542388916016 (Prediction Confidence) <b>**BLEU Score:**</b> 0.32425127038259477 (Response Quality) <b>**Response Time:**</b> 0.02 seconds (Efficiency)</p>

Fig. 3. Qualitative Analysis of the generated responses comparison of the other two models with Python Educational Support Assistant (PESA-8B-FT), demonstrating its capability to provide accurate and contextually relevant answers..

studies validate the practical benefits of deploying such an advanced model in educational settings, highlighting its potential to revolutionize the way programming is taught and learned.

## V. DISCUSSION AND CONCLUSION

### A. Discussion

This study addresses two primary research questions - (1) How effectively can LLM-based virtual assistants improve learning outcomes in Python programming? and (2) What are the limitations of such technology in educational settings, particularly concerning scalability, adaptability, and the depth of educational interaction?

#### Effectiveness of LLM-Based Virtual Assistants in Improving Learning Outcomes

The quantitative and qualitative evaluations indicate that PESA-8B-FT significantly enhances the quality of responses in Python programming education. The model's ability to deliver contextually relevant, precise, and understandable responses underscores its potential to transform educational methodologies. Our results show that PESA-8B-FT achieves lower perplexity and higher BLEU scores compared to other models, suggesting superior performance in generating coherent and contextually appropriate responses. Feedback from participants also supports these findings, with users highlighting the model's clarity and relevance in response generation.

However, while the model shows promise in improving the learning experience, claiming significant improvements in overall learning outcomes requires more extensive and longitudinal studies. Compared to traditional syntax repair tools, PESA-8B-FT not only corrects errors but also provides detailed explanations and personalized feedback, enhancing both the learning experience and coding accuracy.

#### Limitations of LLM-Based Virtual Assistants in Educational Settings

Despite its effectiveness, several limitations must be addressed to realize the full potential of LLM-based virtual assistants in educational settings. Scalability is a primary concern; while PESA-8B-FT performed well in controlled environments, scaling this technology to larger and more diverse student populations presents significant challenges. The computational resources required for real-time interactions and continuous model updates are substantial, potentially limiting widespread adoption in resource-constrained educational institutions.

Adaptability is another critical issue. Although PESA-8B-FT was fine-tuned using a comprehensive corpus of Python programming materials, its ability to adapt to new, unseen problems depends on the breadth and quality of the training data. In rapidly evolving fields like programming, where new techniques and tools frequently emerge, maintaining the model's relevance and accuracy necessitates ongoing updates and retraining, which can be resource-intensive.

Furthermore, the depth of educational interaction provided by LLM-based virtual assistants remains limited compared to human educators. While PESA-8B-FT excels in delivering accurate and contextually relevant responses, it lacks the ability to engage in complex, multi-turn dialogues that explore deeper learning pathways and foster critical thinking. Human educators bring empathy, intuition, and pedagogical expertise that LLMs cannot yet replicate. Therefore, PESA-8B-FT is best utilized as a supplementary tool rather than a replacement for human instructors.

### B. Conclusion

PESA-8B-FT not only meets but exceeds the demands of modern educational tools, offering a robust solution for personalized and interactive learning experiences in Python programming education. The integration of ORPO Fine-

Tuning with the Llama 3 model has resulted in a virtual assistant capable of delivering contextually relevant, precise, and understandable responses, thereby significantly enhancing the educational experience.

The quantitative analysis, supported by perplexity and BLEU scores, establishes PESA-8B-FT's superiority over other models in generating accurate and coherent text. Although the response time is slightly longer than that of DialoGPT-medium, it remains efficient enough for real-time educational applications, balancing speed with the quality of responses.

Qualitative feedback and case studies further validate the practical benefits of our model. Participants appreciated the accuracy and contextual relevance of the responses, which contributed to a better understanding of programming concepts. The ability of PESA-8B-FT to provide detailed, easy-to-understand explanations and facilitate collaborative learning underscores its potential to revolutionize programming education.

In conclusion, PESA-8B-FT sets a new standard for AI-driven educational support by significantly enhancing the quality of interaction and response generation. Future research should explore further refinements to response times, expand the training dataset to cover a broader range of programming scenarios, and investigate the model's application across different educational domains. Future research should include a longitudinal study spanning at least two academic years, tracking students' performance, engagement, and retention rates. Additionally, the study should involve diverse educational settings to assess the scalability and adaptability of the model. By addressing these areas, the full potential of LLM-based virtual assistants like PESA-8B-FT in transforming education can be realized.

## REFERENCES

- [1] Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S. Yu, and Qingsong Wen. 2024. Large Language Models for Education: A Survey and Outlook. In arXiv preprint arXiv:2403.18105.
- [2] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2024. A Comprehensive Overview of Large Language Models. In arXiv preprint arXiv:2307.06435.
- [3] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.
- [4] Hao Sun. 2023. Offline prompt evaluation and optimization with inverse reinforcement learning. arXiv preprint arXiv:2309.06553 (2023).
- [5] Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. 2023. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. In *Workshop on Efficient Systems for Foundation Models@ ICML2023*.
- [6] Claudia M König, Christin Karrenbauer, and Michael H Breitner. 2023. Critical success factors and challenges for individual digital study assistants in higher education: A mixed methods analysis. *Education and Information Technologies* 28, 4 (2023), 4475–4503.
- [7] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.
- [8] Wenhui Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. arXiv preprint arXiv:2211.12588 (2022).
- [9] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023a. Pal: Program-aided language models. In *International Conference on Machine Learning*. PMLR, 10764–10799.
- [10] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023a. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. arXiv preprint arXiv:2308.08155 (2023).
- [11] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168 (2021).
- [12] Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, et al. 2024b. Solving Challenging Math Word Problems Using GPT-4 Code Interpreter with Code-based Self-Verification. In *The International Conference on Learning Representations*.
- [13] Xiaowu Zhang, Xiaotian Zhang, Cheng Yang, Hang Yan, and Xipeng Qiu. 2023b. Does Correction Remain An Problem For Large Language Models? arXiv preprint arXiv:2308.01776 (2023).
- [14] Tung Do Viet and Konstantin Markov. 2023. Using Large Language Models for Bug Localization and Fixing. In *2023 12th International Conference on Awareness Science and Technology (iCAST)*. IEEE, 192–197.
- [15] Majeed Kazemitabaar, Xinying Hou, Austin Henley, Barbara Jane Ericson, David Weintrop, and Tovi Grossman. 2023. How novices use LLM-based code generators to solve CS1 coding tasks in a self-paced learning environment. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*. 1–12.
- [16] Jaromir Savelka, Arav Agarwal, Marshall An, Chris Bogart, and Majd Sakr. 2023. Thrilled by your progress! Large language models (GPT-4) no longer struggle to pass assessments in higher education programming courses. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 1*. 78–92.
- [17] Shen, W., Xu, T., Li, H., Zhang, C., Liang, J., Tang, J., Yu, P. S., & Wen, Q. (2024). Large Language Models for Education: A Survey and Outlook. arXiv preprint arXiv:2403.18105.
- [18] Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 311-318). Retrieved from <https://www.aclweb.org/anthology/P02-1040>
- [19] Zhang, X., Zhang, X., Yang, C., Yan, H., & Qiu, X. (2023). Does Correction Remain a Problem for Large Language Models? arXiv preprint arXiv:2308.01776. Retrieved from <https://arxiv.org/abs/2308.01776>