Testing Fine-Tuned Large Language Models for Power System Analysis

Nikola Slavchev, Eric M. Keller, Jonathan M. Snodgrass, Thomas J. Overbye
Dept. of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA
Emails: nislavch@tamu.edu, erickeller@tamu.edu, snodgrass@tamu.edu, overbye@tamu.edu

***Abstract*—Power system studies often involve large sets of operational data, simulation results, and technical documents that are not always straightforward to interpret quickly during disruptive events. This work presents a local language-model environment that uses retrieval-augmented search and targeted fine-tuning on power systems material to improve the clarity and relevance of model responses. The system indexes CSV and PDF data using a FAISS vector store for fast retrieval and can also run user-supplied Python scripts when a prompt requires additional analysis. A planner module determines which files or tools are necessary based on the user's question, enabling natural-language queries to trigger analytical workflows. The framework operates fully offline, supporting environments where data sensitivity and continuity are crucial. Early testing suggests that the approach can reduce manual effort and support situational awareness in hazard-resilient planning tasks.**

***Index Terms*—Power system planning, grid resilience, hazard analysis, retrieval-augmented generation (RAG), large language models, domain-specific fine-tuning, FAISS, automated analytical workflows**

I. Introduction

THE rapid advancement of Artificial Intelligence in recent times has opened new possibilities to predict grid stress and utilization, manage real-time grid operations, and enhance grid resilience. With the increasing frequency of extreme weather events and evolving weather patterns, maintaining an uninterrupted power supply has become more challenging and often requires swift and accurate decision-making. Conventional tools often require significant manual labor and respond slowly to sudden or unexpected events. The addition of renewable energy sources to the grid introduces variability, which further complicates grid operations and management. This necessitates a more adaptive, data-driven, and time-efficient mode of operation that leverages AI models to anticipate, prevent, and recover from grid interruptions.

This paper presents a unified AI environment for large language model-enhanced hazard-resilient power system planning. While existing generic LLMs offer broad reasoning capabilities, they lack specific knowledge of power system protection schemes and grid topology, often leading to hallucinations. Furthermore, cloud-based solutions pose significant security risks for critical infrastructure data. To bridge this gap, our framework integrates retrieval-augmented generation (RAG) and domain-specific fine-tuning within a fully offline, local compute environment. The strategy seeks to improve situational awareness and support broader objectives for resilient and secure energy systems by automating analytical workflows and simulation tasks using natural language reasoning, therefore ensuring both domain accuracy and data sovereignty.

II. Background and Motivation

A. Challenges in Hazard-Resilient Power System Planning

Power systems are responsible for providing electricity to data centers, businesses, and consumers, and must maintain high levels of reliability to ensure normal operations. Hazard-resilient planning involves systematically anticipating severe weather changes, their effects on the power grids, and how to recover from severe disruptions. Extreme weather events, such as hurricanes, wildfires, heatwaves, droughts, and ice storms, can have a significant impact on the grid and are often costly. Other challenges in grid resilience include failure or aging of equipment, uncertainty in renewable energy generation, and cyber or physical attacks.

B. Role of Language Models and Retrieval-Augmented Search

Operating power grid systems involves navigating complex manuals, understanding protocols for different scenarios, and responding to highly dynamic changes in grid outputs. Large volumes of operational and planning data are used to make real-time decisions. Such time-critical decisions could include adjusting generator dispatch, coordinating restoration after outages, issuing operating limits to maintain stability of voltage and frequency, and rerouting power flows to reduce load on overloaded lines. Large Language Models (LLMs) can quickly process natural language input, analyze data, identify patterns, and generate solutions efficiently. In combination with retrieval-augmented generation (RAG), LLMs can retrieve, read, and analyze relevant stored data, power grid simulation results, and real-time telemetry data. RAG enables natural language processing models to reference the same data that grid operators use to make decisions, thereby increasing the efficiency and consistency of decision-making.

C. Need for Local and Domain-Specific AI Tools

The storage and handling of power system data must be done securely and reliably. Grid data often contains sensitive information that, if compromised, can present a serious public risk. Power grids are critical infrastructure, and therefore, data must be used safely within an AI environment. As a result, LLMs running on local, offline compute are well-suited for the use case of power grid analysis. They provide a controlled environment where data is stored exclusively on local server hardware. Often, these LLMs are open-source and can be fine-tuned with domain-specific data. A unified AI environment for resilient power grid planning would include a specialized, fine-tuned reasoning LLM combined with the ability to retrieve and analyze stored data, run program scripts, and derive conclusions.

D. Related Work

Recent advances in language-model research relate to the methods used in this work. Retrieval-augmented generation improves factual accuracy by grounding model outputs in retrieved document segments [1]. Embedding models further support this process by mapping text into dense vectors for efficient similarity search; for example, Sentence-BERT provides high-quality transformer-based sentence embeddings [2]. These developments align with the hardware-accelerated vector search capabilities of FAISS [3], which we adopt in our system. Together, these works form the foundation for the RAG approach used in this paper.

Recent research is shifting from simple retrieval to fine-tuning models specifically for power systems. A 2025 review by Mirshekali et al. [4] breaks down this landscape (Table 3) and shows that using Low-Rank Adaptation (LoRA) on open-source models is a standard approach. They point to frameworks like RE-LLaMA and LFLLM [4] which use this method. This supports our decision to use LoRA for our own domain adaptation. In parallel, we see new agent systems like X-GridAgent [5]. That project explicitly integrates retrieval-augmented generation (RAG) to automate grid tasks, which mirrors our own use of RAG for data retrieval. Similarly, models like WildfireGPT [6] show the value of tailoring LLMs to specific hazard contexts. However, while these systems have powerful reasoning, they typically rely on external cloud APIs. Our implementation takes a different approach by running the entire pipeline and retrieval system offline. This isolation is critical for mitigating agentic threats such as tool misuse and unexpected remote code execution, as highlighted in the recent OWASP Agentic AI Threat Model [7].

III. System Architecture and Workflow

A. Retrieval-Augmented Generation Environment

The proposed system is implemented as a local retrieval-augmented generation platform. At the center of the design is a GPU-resident large language model (LLM) that handles natural language reasoning. In this work, open-source LLaMA-based models were used [8], allowing the system to be deployed on local hardware without reliance on external APIs. A FastAPI interface [9] manages user interaction, file I/O, and controlled execution of analysis tasks. A planner module interprets each user query and identifies the relevant datasets or scripts. CSV tables and PDF documents are indexed in a FAISS vector database [3] to enable fast similarity-based retrieval. Small adapter modules then convert the returned data into short text snippets that the model can reference while forming a response.

The workflow proceeds as follows. A user submits a natural-language question. The planner analyzes the question and selects the required data sources or tools. The retrieval layer returns the most relevant document segments or CSV entries. If the query involves computation, the system can execute user-provided Python scripts in a sandboxed environment. The LLM then generates a final response based on both the retrieved context and the user query.

Since the system operates entirely on local hardware, all data remains within a controlled environment, aligning with the security and reliability requirements of hazard-resilient power system planning. Running the LLM, retrieval processes, and execution modules locally prevents external data exposure and supports use in organizations with strict compliance constraints. The design is modular, allowing different models, embedding backends, or retrieval mechanisms to be substituted as needed.

B. Domain-Specific Fine-Tuning of the LLM

To make the language model more familiar with power-system concepts, diagrams, and the style of technical explanations found in textbooks, the base LLaMA-3.2-3B-Instruct model [8] was fine-tuned using a custom dataset. The dataset was built from textbook page screenshots covering a wide range of topics in power systems [10]. These images were processed using an image-understanding model (GPT-4o), which generated 50 high-quality prompt–completion training pairs for each batch of pages. The generation prompt required the model to use only the information visible in the images, explain diagrams in natural language, avoid page numbers and figure labels, and write every training pair as a clear, self-contained explanation. This led to a diverse collection of well-structured examples drawn directly from the textbook content.

Each pair was converted into the standard LLaMA instruction format, "[INST] prompt [/INST] completion." The model received both parts as input, but only the completion (the assistant's answer) was used for learning. The prompt tokens were ignored in the loss calculation so the model would learn to produce correct explanations without trying to memorize the questions themselves. The tokenizer was also updated with additional role tokens and a proper padding token to improve stability during training.

Fine-tuning was performed using LoRA, a lightweight method that adds a small number of trainable parameters to the model instead of updating all weights [11]. In our implementation, LoRA was applied through the PEFT library [12], which provides parameter-efficient fine-tuning methods for transformer models. This made it possible to fine-tune the model efficiently on a single workstation GPU using FP16 precision. Training was run for two epochs with a moderate learning rate. After fine-tuning, the model showed improved understanding of power-system terminology and the types of explanations commonly found in engineering textbooks. The

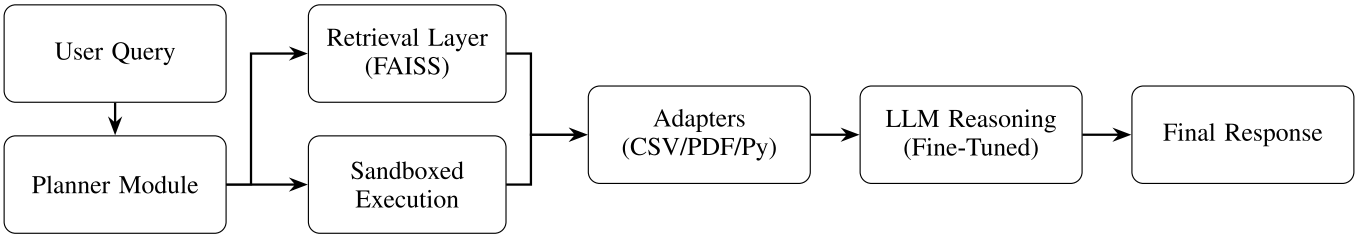

Fig. 1. System workflow: The planner dynamically routes queries to either the retrieval layer or the sandbox execution environment before synthesis.

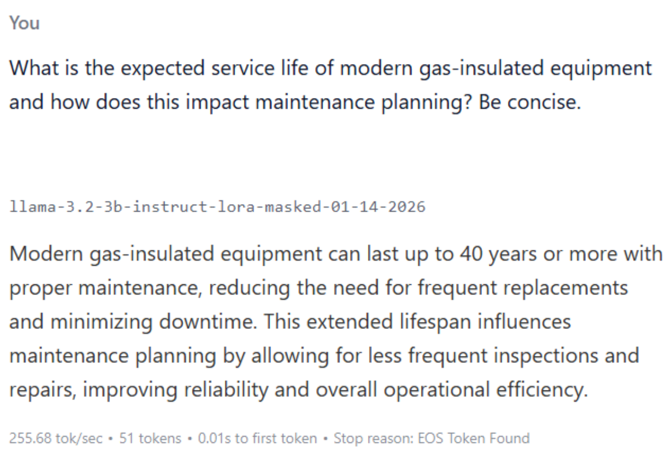

Fig. 2. Example response from the fine-tuned LLaMA-3.2-3B-Instruct model on a power-systems question.

resulting adapter weights and tokenizer were then integrated into the retrieval-augmented system.

IV. Case Study / Example Usage Scenario

As a simple demonstration, we evaluated the workflow using both the base and fine-tuned models to interpret a PowerWorld simulation script written with the ESA Python API. The script connects to the PowerWorld simulator, loads the synthetic 40-bus Hawaii system test case [13], and executes both a power flow and a dynamic simulation. Upon completion, it extracts and prints a "System Health Report" containing total generation versus load, calculated system losses, active generator dispatch, and filtered lists for critical line loadings ($> 80\%$) and voltage violations (outside 0.95–1.05 pu). Since the current system does not pass parameters to scripts, the planner's role is to recognize that the request requires executing a specific Python file. Once selected, the sandbox runs the script locally, and the console output is captured and returned to the LLM as context.

Using a prompt such as "Run and analyze the Hawaii grid case Python script and synthesize key insights," the system correctly identified the script, executed it, and returned its raw output. Both models attempted to synthesize this data. As detailed in the results section, the models identified the generation surplus, quantified the system losses (derived from the script's generation and load totals), and highlighted specific critical transmission lines (see Fig. 9 and Fig. 10).

We also tested specific follow-up questions, including "What is the total active power at bus 33?" and "Are there any voltage violations in the system?" which required the models to parse the script's tabular output to find precise values or confirm adherence to the defined limits. Figure 3 and Figure 4 demonstrate the responses from the base model, while Figure 5 and Figure 6 show the equivalent outputs from the fine-tuned model. These examples illustrate that both models can extract specific data points, such as the 27.60 MW active power at bus 33 or the lack of voltage violations outside the 0.95–1.05 pu range, from the raw simulation results.

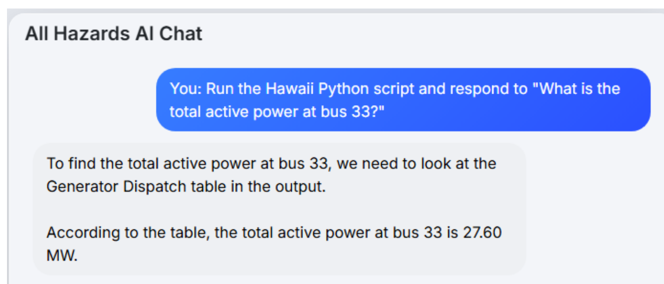

Fig. 3. Base model response to a specific query regarding active power at bus 33.

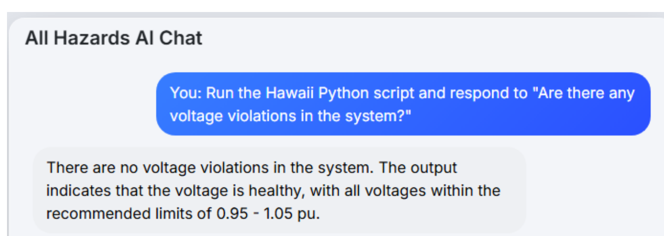

Fig. 4. Base model response confirming the absence of voltage violations based on simulation output.

This example shows how the environment can help streamline basic study tasks by running simulation scripts and producing quick, readable summaries of the results. Although the script interface is currently limited to fixed inputs, the workflow effectively supports operational queries without requiring the user to manually inspect raw log files.

V. Results and Discussion

During initial testing, we observed several limitations in how the model generated its answers. Many responses ended

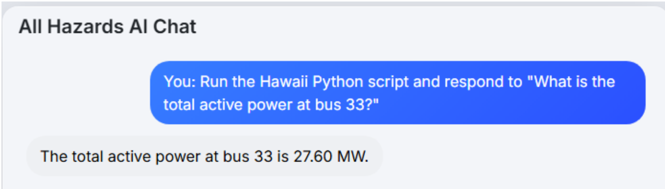

Fig. 5. Fine-tuned model response identifying the total active power at bus 33.

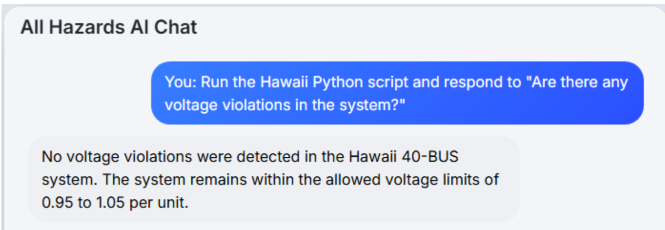

Fig. 6. Fine-tuned model response confirming system health regarding voltage limits.

up having similar lengths and structures, and some were shorter or less detailed than expected. This behavior was mainly linked to the training data, since the textbook-derived examples followed a fairly consistent format. As a result, the model tended to mirror that style. Increasing the variety of examples and including different types of prompts should help reduce this effect in future versions of the dataset.

Figure 7 shows the training loss across the two epochs of LoRA fine-tuning. The loss decreases smoothly and then levels off, which indicates stable training and that the adapter layers were able to learn useful patterns from the instructional pairs. The final loss value, which settled around 1.9, is reasonable given both the size of the dataset and the fact that only the answer portion of each example was used for the loss calculation. The training was performed using meta-llama/Llama-3.2-3B-Instruct [8] and ran efficiently on workstation hardware, since LoRA only introduces a small number of additional trainable parameters [11], [12].

To evaluate the retrieval pipeline, we ran the same workflow that is used during normal operation. Figure 8 illustrates an example run of the RAG framework using the base Llama-3-70B-Instruct model [8] along with the Qwen3-Embedding-0.6B embedding model [14]. In this test, the system successfully retrieved relevant text from a locally stored PDF and supplied the context to the model. The RAG system runs fully offline and was tested on a multi-GPU setup ($2\times$ NVIDIA H100), where the end-to-end retrieval time was typically around 0.5 seconds. Of that total, the FAISS similarity search accounted for less than 0.12 seconds [3]. These measurements show that the indexing, embedding, adapter modules, and planner logic operate reliably and without noticeable latency to the user.

After integrating the fine-tuned model into the same environment, we evaluated how well it could answer power-system questions without relying on retrieval. Figure 2 shows an example response generated by the fine-tuned Llama-3.2-3B-Instruct model. The answer is noticeably clearer and more aligned with the explanatory style found in engineering textbooks compared to responses from the base model. For quantitative evaluation, we used a small held-out test set created by aggregating all prompt and completion pairs and randomly holding out 10% of the data before training, ensuring these examples were completely removed from the training set. Using this internal split, the fine-tuned model achieved a BERTScore-F1 of 0.7735 [15] and a masked perplexity of approximately 7.16 on the full held-out test set. These results indicate strong domain adaptation: the BERTScore suggests high semantic alignment (exceeding the 0.70–0.75 range typically cited for high-quality generation), while the single-digit perplexity reflects predictive fluency (well below the <10 threshold often used for converged conversational models). Although the test samples come from the same overall source as the training data, these results still show that the model learned domain-specific terminology and explanation patterns effectively.

In addition to quantitative metrics, we compared the qualitative performance of the base model versus the fine-tuned model when integrated with the RAG pipeline. Both models were tasked with interpreting the raw output of a Python contingency analysis script (as described in Section IV). Figure 9 shows the response from the base LLaMA-3.2-3B model. While factually correct, the base model tends to list the script outputs mechanically ("The generator dispatch table shows...", "The critical lines report highlights..."). In contrast, Figure 10 displays the response from the fine-tuned model on the same task. The fine-tuned model synthesizes the information more effectively, adopting a professional engineering tone ("...at the cost of increased system losses," "future applications could benefit from..."). This comparison highlights that while RAG provides the correct facts to both models, the domain-specific fine-tuning improves the presentation and synthesis of that data, making it more immediately useful for planning tasks.

To mitigate risks associated with misleading or hallucinated conclusions, which is a critical concern in power system planning, the system employs a "separation of concerns" architecture. We observed that analytical hallucinations, where models invent plausible but incorrect numerical data, are effectively prevented by offloading all computations to the sandboxed execution environment. The LLM acts solely as an interpreter of deterministic script outputs, such as PowerWorld results, rather than generating them. For textual queries, the system limits the model's scope to trusted and locally stored documents, significantly reducing the ingestion of noisy or conflicting information compared to open-web retrieval. While the current prototype relies on vector similarity scores to filter irrelevant context, future work will incorporate explicit consistency checks to flag contradictions between retrieved document segments before an answer is synthesized.

Beyond these comparative evaluations, the system was also evaluated on several practical tasks that reflect real power-grid analysis workflows. These included summarizing out-

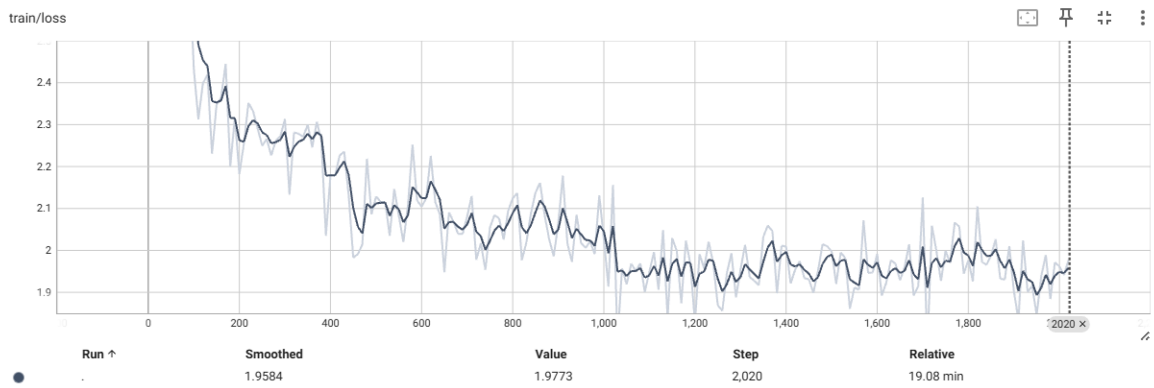

Fig. 7. Training loss convergence over 2,020 update steps (2 epochs) of LoRA fine-tuning.

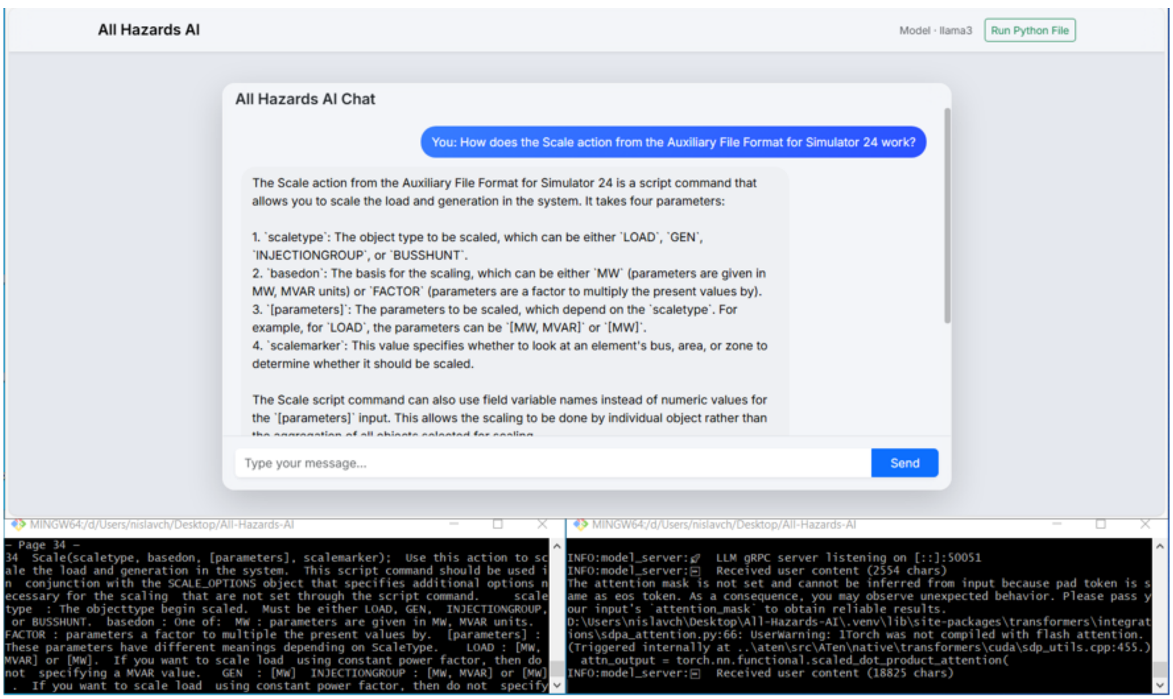

Fig. 8. RAG framework pipeline running with base LLaMA-3-70B-Instruct and Qwen3-Embedding-0.6B, demonstrating retrieval from a locally stored PDF.

age information from PDF documents, running user-supplied PowerWorld or post-processing scripts through the sandboxed execution environment, and querying CSV logs for abnormal operating conditions. In each case, the combined fine-tuned LLM and RAG pipeline was able to retrieve relevant information, interpret it, and provide a structured explanation.

Overall, the results indicate that the fine-tuned model improves the clarity and accuracy of responses to power-system questions, while the RAG system ensures that answers can be grounded in real documents and simulation data when needed [1]. The current system is an early-stage prototype, and future work should focus on expanding the training data with more diverse examples, testing the model on additional document types, and evaluating performance on real planning studies. Even with these limitations, the early findings suggest that combining a fine-tuned model with a local retrieval pipeline can provide a practical and useful tool for supporting hazard-resilient power-system analysis.

VI. Conclusion and Future Work

This paper presented a local AI environment that integrates retrieval-augmented generation with a fine-tuned language model to support hazard-resilient power-system planning. The initial results show that the system can provide clearer explanations and more efficient access to study data while keeping all information local for security-sensitive applications.

The current system has a few practical limitations that reflect its early-stage development. The fine-tuning dataset is still relatively small and relies mainly on textbook-derived examples, which limits the range of situations the model has seen. Generating the prompt–completion pairs with GPT-4o

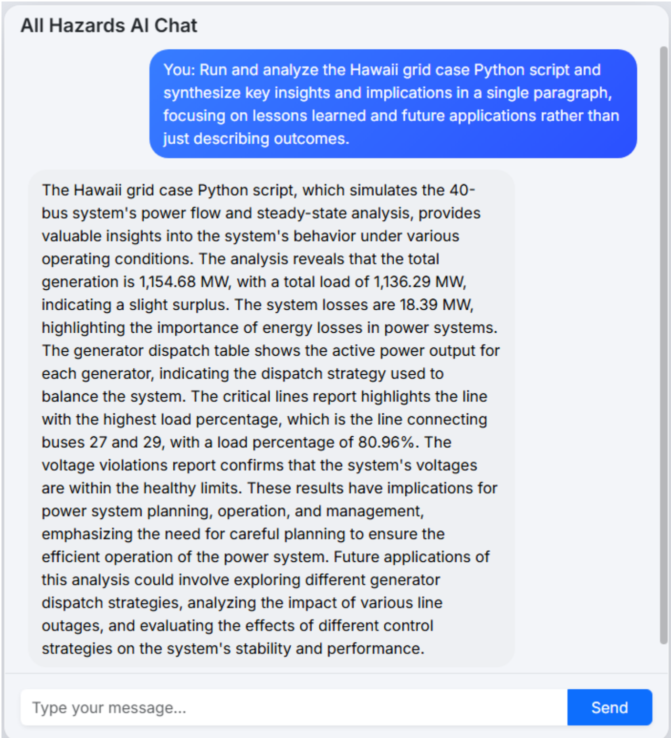

Fig. 9. Response from the base LLaMA-3.2-3B-Instruct model utilizing the RAG framework to analyze Python script output.

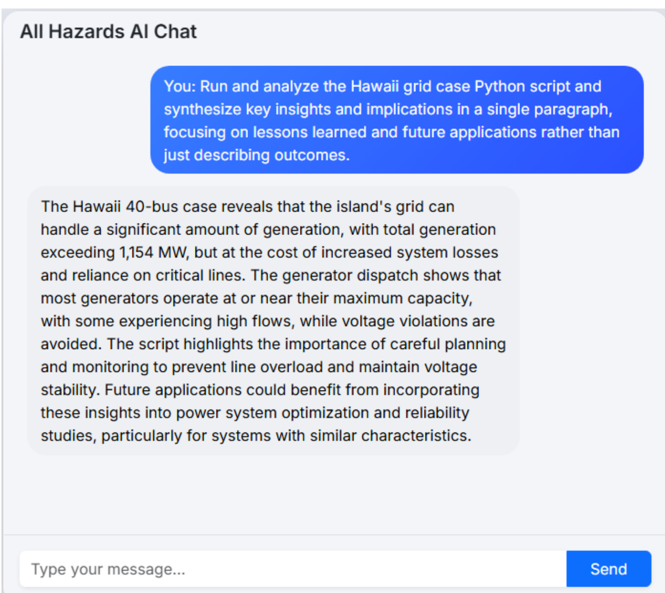

Fig. 10. Response from the fine-tuned LLaMA-3.2-3B-Instruct model utilizing the RAG framework. The model demonstrates improved synthesis and domain-specific tone compared to the base model.

remains a partly manual process and is not yet optimized for larger-scale or repeated data collection. Evaluation has also been limited to the internal test split used for reporting BERTScore (0.7735) and masked perplexity ($\approx$ 7.16), along with a set of qualitative tests on representative planning questions. These constraints do not affect the feasibility demonstrated in this work, but they highlight opportunities for improving data diversity, automating the training pipeline, and expanding validation in future iterations.

Future work will focus on expanding system capabilities and improving scalability. Planned developments include support for multi-user operation, stronger sandboxed script execution, and further optimization of the retrieval pipeline, including migration from FAISS to newer vector-search backends such as Qdrant or pgvector. Additional fine-tuning on long-form scientific material and more diverse Q&A examples is expected to improve the model's reasoning depth. A unified interface with cited answers is also planned to enhance usability for grid-planning workflows.

References

[1] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open-domain question answering," in *Proc. 16th Conf. Eur. Chapter ACL (EACL)*, 2021, pp. 874–880. [Online]. Available: https://arxiv.org/abs/2007.01282

[2] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *Proc. 2019 Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2019, pp. 3982–3992.

[3] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2021.

[4] H. Mirshekali, M. R. Shadi, F. G. Ladani, and H. R. Shaker, "A review of large language models for energy systems: Applications, challenges, and future prospects," *IEEE Access*, vol. 13, pp. 163 162–163 188, 2025.

[5] Y. Wen and X. Chen, "X-GridAgent: An LLM-powered agentic AI system for assisting power grid analysis," *arXiv preprint arXiv:2512.20789*, 2025. [Online]. Available: https://arxiv.org/abs/2512.20789

[6] Y. Xie, B. Jiang, T. Mallick, J. D. Bergerson, J. K. Hutchison, D. R. Verner, J. Branham, M. R. Alexander, R. B. Ross, Y. Feng, L. Levy, W. J. Su, and C. J. Taylor, "WildfireGPT: Tailored large language model for wildfire analysis," *arXiv preprint arXiv:2402.07877*, 2025, updated Apr. 2025. [Online]. Available: https://arxiv.org/abs/2402.07877

[7] OWASP GenAI Security Project, "Agentic AI - threats and mitigations, version 1.1," Feb. 2025, accessed: Jan. 14, 2026. [Online]. Available: https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/

[8] A. Grattafiori *et al.*, "The LLaMA 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[9] S. Ramírez, "FastAPI documentation," https://fastapi.tiangolo.com/, 2024, accessed: Feb. 18, 2025.

[10] J. D. Glover, M. S. Sarma, and T. J. Overbye, *Power System Analysis and Design*, 5th ed. Stamford, CT, USA: Cengage Learning, 2012.

[11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022. [Online]. Available: https://arxiv.org/abs/2106.09685

[12] S. Mangrulkar *et al.*, "PEFT: State-of-the-art parameter-efficient fine-tuning methods," GitHub repository, 2022. [Online]. Available: https://github.com/huggingface/peft

[13] Texas A&M University Electric Grid Test Case Repository, "Synthetic Hawaii 40-bus system," https://electricgrids.engr.tamu.edu/hawaii40/, accessed: 2026-01-14.

[14] A. Yang, B. Yang, B. Hui *et al.*, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024. [Online]. Available: https://arxiv.org/abs/2407.10671

[15] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating text generation with BERT," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020. [Online]. Available: https://arxiv.org/abs/1904.09675