Towards a Domain-Specific AI Framework for Power System Simulation Analysis

Hwiyoon Kim, Eric Keller, Jonathan M. Snodgrass, Thomas J. Overbye
Department of Electrical and Computer Engineering
Texas A&M University
College Station, TX
{hwiyoonkim, erickeller, snodgrass, overbye}@tamu.edu

***Abstract*—This paper presents a prototype framework for developing domain-specific artificial intelligence applications in power systems analysis. An automated pipeline is demonstrated that transforms power system simulation files into structured databases, generates large-scale question-answer datasets, and implements retrieval-augmented generation with parameter-efficient fine-tuning. The system successfully converts power flow case files into SQLite databases, generates over 1000 natural language question-answer pairs grounded in simulation data, and enables interactive querying through a combination of retrieval-augmented generation and Low-Rank Adaptation fine-tuning of language models. The framework addresses the critical gap between specialized power systems knowledge encoded in simulation tools and the need for accessible, explainable AI-driven decision support systems in the energy sector.**

***Index Terms*—Power systems, artificial intelligence, natural language processing, retrieval-augmented generation, fine-tuning, domain-specific knowledge**

I. Introduction

THE electric power industry faces increasing complexity in grid operations, planning, and reliability assessment. Modern power systems analysis relies heavily on specialized simulation tools, which enable engineers to model power flows, assess system stability, and plan infrastructure expansions. However, the steep learning curve associated with these tools and the specialized knowledge required to interpret their outputs present significant barriers to broader adoption and automated decision-making systems.

Recent advances in large language models (LLMs) have demonstrated remarkable capabilities in understanding and generating human language across diverse domains [1], [2]. However, general-purpose LLMs lack the specialized knowledge, terminology, and topological understanding required for effective power systems analysis. This fundamental limitation motivates the development of domain-specific AI frameworks that can bridge the gap between simulation tools and natural language interfaces.

The motivation for this work stems from three key observations. First, power systems engineers currently spend considerable time manually extracting, interpreting, and communicating insights from simulation results. Second, the knowledge encoded in simulation tools remains largely inaccessible to non-specialists and automated systems. Third, the growing complexity of modern grids with increased renewable integration and distributed resources demands more intelligent, explainable decision-support systems that can reason about power system behavior [3].

The work presented here has been done using PowerWorld Simulator [4].

A. Research Objectives

This project aims to develop a scalable, replicable framework with four primary objectives. First, to automate the transformation of input `.pwb` files into structured, queryable knowledge bases. Second, to generate large-scale, domain-specific question-answer datasets grounded in actual simulation data. Third, to implement retrieval-augmented generation to enable accurate, context-aware responses to natural language queries about power systems. Finally, to explore parameter-efficient fine-tuning techniques using Low-Rank Adaptation to enhance model performance while maintaining computational efficiency.

The significance of this work extends beyond power systems to any domain requiring the integration of specialized simulation tools with natural language AI interfaces. The framework provides a replicable methodology for converting domain-specific data into training datasets and developing intelligent assistants capable of interactive, explainable decision support.

II. Background and Related Work

A. Power System Simulation Tools

Modern power systems analysis software provides comprehensive capabilities for power flow analysis, contingency screening, and system planning. The software offers both graphical and programmable interfaces, supporting various export formats including auxiliary (AUX) files and comma-separated values (CSV). However, converting these exports into structured, queryable representations suitable for machine

Funding was provided in part by the Department of Energy (DOE) Office of Cybersecurity, Energy Security, and Emergency Response (CESER) through project DE-CR0000067. This work is also part of the "Electric Grid Resilience" Project (60NANB24D210) funded by the National Institute of Standards and Technology of US Department of Commerce. The views expressed herein do not necessarily represent the views of the U.S. Department of Energy or the United States Government.
Copyright © 2026 IEEE. Personal use of this material is permitted. This paper was presented at the Texas Power and Energy Conference (TPEC), College Station, TX, February 2026.

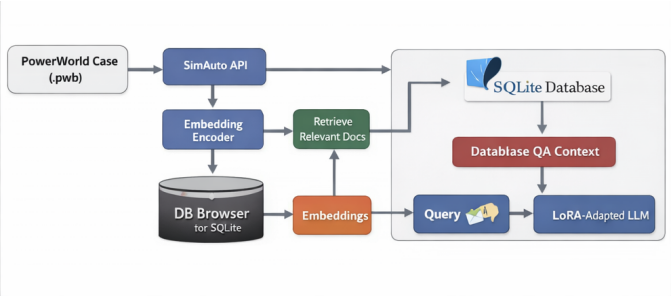


Fig. 1. Retrieval-augmented generation pipeline used in the proposed framework. The system converts `.pwb` cases to SQLite, builds embeddings, retrieves relevant evidence, and provides grounded context to the LoRA-adapted LLM for response generation.

learning applications typically requires significant manual effort and domain expertise.

The SimAuto COM API provides programmatic access to simulation functionality, enabling automated case manipulation and data extraction. The framework leverages this API to systematically convert power system cases into structured SQLite databases, preserving the topological and electrical characteristics of the network while making the data readily accessible for downstream AI applications [5].

B. Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) has emerged as a powerful technique for enhancing language model capabilities by injecting external knowledge at inference time [6]. Unlike traditional fine-tuning approaches that embed knowledge into model parameters, RAG systems maintain explicit knowledge bases that can be dynamically queried to provide relevant context for generating responses.

For domain-specific applications such as power systems analysis, RAG offers several critical advantages. First, it significantly improves factual accuracy by grounding model outputs in verified simulation data rather than relying solely on parametric knowledge that may be incomplete or outdated [7]. Second, RAG reduces the risk of hallucination, a common problem where language models generate plausible but incorrect information. Third, RAG systems can seamlessly incorporate updated data without requiring model retraining, making them particularly suitable for dynamic environments where simulation results change frequently.

The implementation combines semantic similarity search using sentence embeddings [8] with a carefully designed retrieval strategy that prioritizes simulation-derived facts. This approach ensures that responses to queries such as "What is the base voltage at Bus 301?" are grounded in actual case data rather than statistical patterns learned during pre-training.

C. Parameter-Efficient Fine-Tuning with LoRA

Low-Rank Adaptation (LoRA) represents a breakthrough in efficient neural network fine-tuning by introducing trainable low-rank decomposition matrices into frozen pre-trained model weights [9]. Instead of updating all parameters during fine-tuning, LoRA adds pairs of rank-decomposition matrices to existing weights, dramatically reducing the number of trainable parameters while maintaining performance comparable to full fine-tuning.

The mathematical foundation of LoRA rests on the observation that updates to neural network weights during fine-tuning often have low intrinsic rank [10]. For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA modifies the forward pass computation as:

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{1}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ are trainable low-rank matrices with rank $r \ll \min(d, k)$. The matrix B is initialized to zero so that $\Delta W = BA$ equals zero at the start of training, while A is initialized with a random Gaussian distribution.

To control the magnitude of adaptations, LoRA scales the updates by α/r where α is a hyperparameter:

$$h = W_0 x + \frac{\alpha}{r} BAx \tag{2}$$

This scaling allows the learning rate to remain constant when varying the rank r, providing better hyperparameter tuning stability. The key advantage is that W_0 remains frozen, and only the significantly smaller matrices A and B require gradient computation and storage during training. For a weight matrix of dimension $d \times k$, full fine-tuning requires updating $d \times k$ parameters, whereas LoRA updates only $r \times (d + k)$ parameters, achieving substantial memory and computational savings when $r \ll \min(d, k)$.

For the power systems application, LoRA is applied to local instances of Meta-LLaMA models (3B and 8B parameter variants) using the custom question-answer dataset [2]. Experiments explore different rank configurations ($r = 8$ and $r = 16$) and scaling parameters ($\alpha = 16$ and $\alpha = 32$) to identify optimal configurations that balance adaptation capability with computational efficiency and memory constraints [11].

III. Methodology

A. Automated Dataset Construction Pipeline

The framework implements a fully automated pipeline that transforms raw simulation files into structured knowledge suitable for machine learning applications. The pipeline consists of five sequential stages, each designed to preserve data integrity while progressively enriching the representation. Figure 3 illustrates the interactive web-based interface that facilitates seamless user interaction with the system.

1) File Upload and Validation: The process begins when a user provides a `.pwb` case file through a web interface. The system validates file integrity, checks format compatibility, and extracts basic metadata including file size, creation date, and version information. This validation step prevents downstream errors and ensures that only well-formed cases proceed to subsequent processing stages.

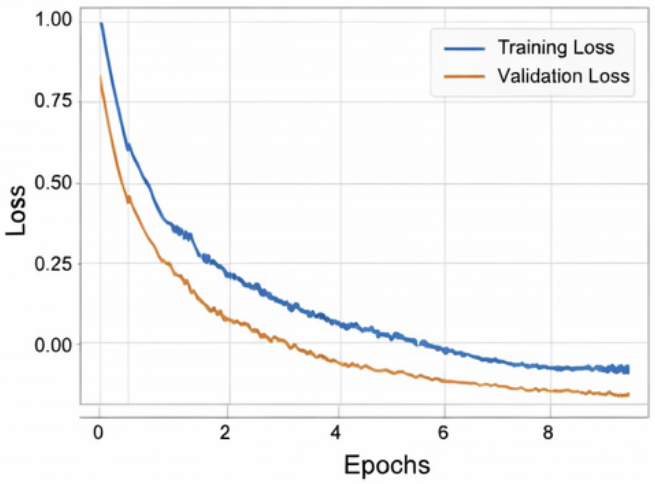


Fig. 2. Representative training and validation loss curves during LoRA fine-tuning on simulation-grounded QA data, illustrating stable convergence.

2) SimAuto Interface: Upon validation, the system invokes the SimAuto COM interface to programmatically open and manipulate the case file. This interface provides low-level access to all case components including buses, generators, loads, branches, transformers, and other network elements. The implementation carefully manages COM object lifecycles to prevent memory leaks and ensure reliable operation across diverse case sizes and complexities.

3) Structured Database Export: The core transformation converts the internal representation into a normalized SQLite database schema. This schema captures the network topology through tables representing buses, generators, loads, and transmission elements, while preserving electrical properties such as voltage magnitudes, power injections, and impedances. The relational structure enables efficient querying and maintains referential integrity through foreign key constraints.

The database schema design, illustrated in Figure 4, prioritizes both completeness and usability. Each network component type maps to a dedicated table with columns corresponding to standard power systems parameters. The relational database structure provides several advantages over flat file formats: (1) it enables complex queries joining multiple tables to answer topological questions, (2) it enforces data integrity through primary and foreign key constraints, (3) it supports efficient indexing for rapid data retrieval, and (4) it provides a standardized SQL interface accessible to multiple programming languages and analysis tools [12].

For instance, the Bus table includes fields for base voltage (`NomKV`), bus name (`BusName`), area assignment (`AreaNum`), and zone designation (`ZoneNum`), as shown in Figure 4. Similarly, the Branch table captures from-bus and to-bus identifiers, resistance, reactance, and thermal ratings. This normalized structure eliminates data redundancy while maintaining the ability to reconstruct complete network topology through table joins.

The SQLite database format offers particular advantages for this application. Unlike client-server database systems such as PostgreSQL or MySQL, SQLite operates as a self-contained, serverless database engine with zero configuration requirements [13]. This makes the framework easily deployable across different computing environments without additional infrastructure. Furthermore, SQLite databases are stored as single cross-platform files, facilitating portability and version control integration.

The automated conversion process handles various object types including buses, generators, loads, transmission lines, transformers, shunt devices, and switched shunts. Each object type's properties are extracted via SimAuto API calls and inserted into corresponding database tables using parameterized SQL queries to prevent injection vulnerabilities. The process includes error handling to manage cases with missing or malformed data, logging warnings while ensuring the overall conversion completes successfully.

4) Statistical Summarization: Following database creation, the system generates a statistical summary of the case by executing aggregate SQL queries. This summary includes fundamental metrics such as the total number of buses, generators, loads, and branches, identifies the slack bus, and calculates aggregate quantities including total generation capacity and load demand. These summaries provide quick overviews of case characteristics and serve as input for subsequent question generation.

The statistical module executes queries such as:

```
SELECT COUNT(*) FROM Bus;
SELECT SUM(GenMW) FROM Generator
  WHERE GenStatus = 'Closed';
```

These queries leverage SQL's aggregation capabilities to efficiently compute system-wide statistics without loading entire tables into memory, enabling scalability to large cases with thousands of components.

5) Question-Answer Generation: The final stage generates synthetic yet grounded natural language question-answer pairs that cover diverse aspects of power system knowledge. The generation strategy employs template-based methods augmented with random sampling to ensure coverage of different question types, difficulty levels, and system components [14].

The template library includes patterns for object counting ("How many buses are in the system?"), property queries ("What is the base voltage at Bus X?"), topological questions ("Which generators are connected to Bus Y?"), and analytical queries ("What is the total installed generation capacity?"). Each template instantiation draws from actual case data retrieved via SQL queries, ensuring that every question has a verifiable answer grounded in simulation results.

For a typical case with 1,430 buses and 45 generators, the system generates over 1,000 distinct question-answer pairs. This scale ensures sufficient training data for fine-tuning while maintaining diversity to prevent overfitting. Example question-answer pairs include:

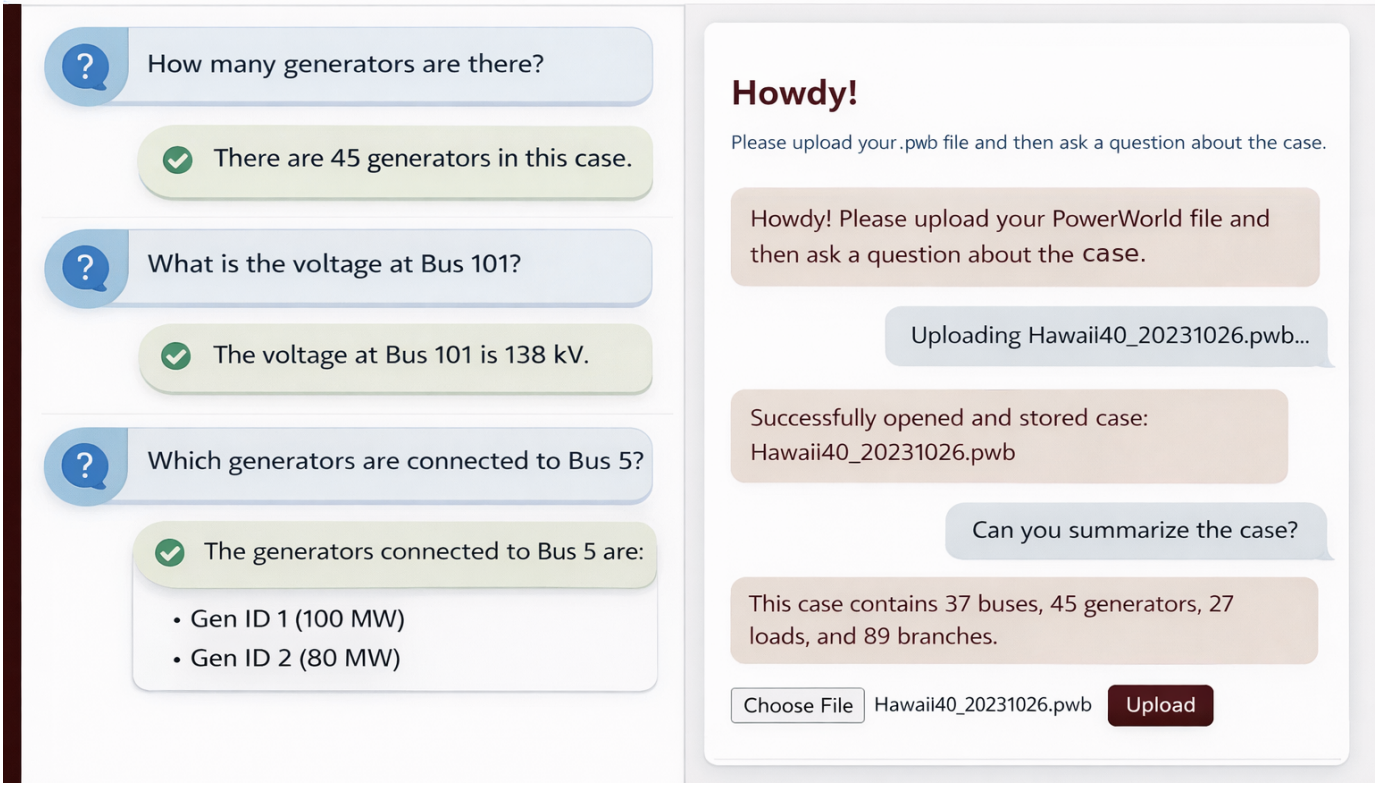
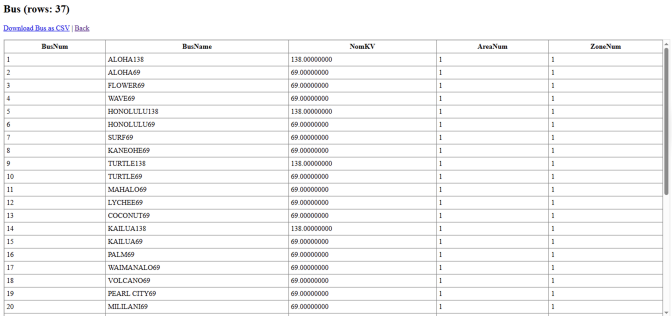


Fig. 3. Web-based conversational interface for power system case analysis. The system accepts `.pwb` files, processes them through the SimAuto API, stores extracted objects into SQLite, and enables natural language querying (e.g., counts, bus voltages, and connectivity queries).



Bus (rows: 37)

Download Bus as CSV | Back

BusNum	BusName	NomKV	AreaNum	ZoneNum
1	ALOHA138	138.00000000	1	1
2	ALOHA69	69.00000000	1	1
3	FLOWER69	69.00000000	1	1
4	WAVE69	69.00000000	1	1
5	HONOLULU138	138.00000000	1	1
6	HONOLULU69	69.00000000	1	1
7	SURF69	69.00000000	1	1
8	KANEOHE69	69.00000000	1	1
9	TURTLE138	138.00000000	1	1
10	TURTLE69	69.00000000	1	1
11	MAHALO69	69.00000000	1	1
12	LYCHEE69	69.00000000	1	1
13	COCONUT69	69.00000000	1	1
14	KAILUA138	138.00000000	1	1
15	KAILUA69	69.00000000	1	1
16	PALM69	69.00000000	1	1
17	WAIMANALO69	69.00000000	1	1
18	VOLCANO69	69.00000000	1	1
19	PEARL CITY69	69.00000000	1	1
20	MILILANI69	69.00000000	1	1

Fig. 4. SQLite database artifact produced by the pipeline (example Bus table view). The exported schema preserves key identifiers and electrical attributes (e.g., `BusNum`, `BusName`, `NomKV`, `AreaNum`, `ZoneNum`) required for deterministic SQL queries and grounded QA generation.

- **Q:** How many buses are in the system?
  A: The case includes 1,430 buses.
- **Q:** What is the slack bus in this case?
  A: Bus 1 is the designated slack bus in this simulation.
- **Q:** What is the base voltage of Bus 301?
  A: Bus 301 operates at a base voltage of 138 kV.

B. Retrieval-Augmented Generation Implementation

The RAG-based chatbot provides an interactive natural language interface for querying power system simulation data. The implementation consists of three tightly integrated components: a Flask web application serving as the user interface, a local LLaMA language model accessed through `llama-cpp-python` for efficient CPU inference, and a memory-efficient retrieval system based on sentence embeddings.

The retrieval component employs the MiniLM-L6-v2 sentence transformer [8] to encode both the question-answer dataset and user queries into dense vector representations. When a user submits a query, the system computes its embedding and retrieves the k most similar question-answer pairs based on cosine similarity. These retrieved examples provide relevant context that the language model uses to generate informed, accurate responses.

Responses are streamed to the user through an event-stream tokenized interface, providing real-time feedback and reducing perceived latency. This streaming approach significantly improves user experience, particularly for longer responses or when the model needs to process substantial retrieved context.

C. LoRA Fine-Tuning Configuration

Fine-tuning adapts the language model to the power systems domain through supervised learning on the generated question-answer dataset. The implementation uses the Hugging Face PEFT (Parameter-Efficient Fine-Tuning) library [15], which provides robust implementations of various adaptation techniques.

TABLE I
SUCCESSFUL QUESTION ANSWERING ON A REALISTIC POWER SYSTEM CASE

ID	Question	Expected Answer	Predicted Answer	EM	F1	Time (s)
1	How many buses are in the system?	37 buses	37 buses	T	1.00	1.98
2	How many generators are there?	45 generators	45 generators	T	1.00	2.01
3	What is the base voltage of Bus 101?	138 kV	138 kV	T	1.00	1.96
4	Which generators are connected to Bus 5?	Gen ID 1, Gen ID 2	Gen ID 1, Gen ID 2	T	1.00	2.04
5	How many loads are in the case?	27 loads	27 loads	T	1.00	1.99
6	How many branches are there?	89 branches	89 branches	T	1.00	2.02

The experimental configuration explores the hyperparameter space along two primary dimensions: rank and alpha scaling. Rank values of 8 and 16 are tested, representing the dimensionality of the low-rank decomposition matrices. Higher ranks provide greater model capacity for learning domain-specific patterns but increase memory requirements and training time. The alpha parameter controls the scaling of learned updates relative to the frozen base weights, affecting how aggressively the model adapts to the new domain.

Training proceeds over multiple epochs using the AdamW optimizer with learning rate scheduling [16]. A validation split monitors for overfitting and early stopping is employed based on held-out performance. The relatively small size of LoRA parameters (typically under 1% of total model parameters) enables rapid experimentation with different configurations while maintaining computational tractability on consumer hardware.

Evaluation employs a held-out test set constructed from power system cases not seen during training. Performance is measured using exact match accuracy for questions with definitive answers (e.g., "How many buses?") and BLEU scores for more open-ended queries requiring natural language generation [17]. This dual evaluation strategy captures both the model's factual accuracy and its ability to produce fluent, coherent responses.

IV. RESULTS AND ANALYSIS

A. System Performance Evaluation

The framework's capabilities are evaluated through systematic testing on diverse power system cases ranging from small tutorial systems with fewer than 50 buses to realistic synthetic networks with over 1,000 buses. The evaluation focuses on three critical aspects: question answering accuracy, response consistency, and computational efficiency.

B. Successful End-to-End Evaluation on a Realistic Case

To demonstrate correct operation of the proposed platform under normal conditions, we first evaluate the system on a realistic power system case containing non-zero network elements. The queries target deterministic and verifiable information directly derived from the exported SQLite database, including component counts, bus attributes, and connectivity relationships. These results provide concrete evidence of successful end-to-end functionality, complementing the stress-test evaluation presented next.

C. Stress-Test Evaluation

Table II presents evaluation results for a representative test case containing zero network elements, a deliberately challenging scenario designed to test the model's behavior when expected answers contradict learned patterns from training data. This edge case reveals important characteristics of model behavior and highlights areas for improvement.

The results demonstrate a systematic bias toward predicting non-zero values even when the correct answer is zero. This behavior likely stems from the training data distribution, where cases with zero elements are extremely rare compared to normal operating conditions. The model learns strong statistical patterns favoring typical system sizes and applies these patterns even when they conflict with specific case data.

Interestingly, the predicted values show remarkable consistency across questions and exhibit F1 scores around 0.86 for most queries, indicating that the model provides coherent, self-consistent responses even when factually incorrect. The absence of hallucination flags (all marked as False) suggests that the model's errors result from learned biases rather than random confabulation, an important distinction for system reliability.

Response times remain consistently around 2 seconds per query regardless of question complexity, demonstrating the efficiency of the implementation. This performance makes the system suitable for interactive use while providing headroom for more complex queries or larger retrieved context windows.

D. Analysis and Discussion

The evaluation results highlight both the promise and current limitations of domain-specific language model applications in power systems. On one hand, the system successfully demonstrates the technical feasibility of converting simulation data into natural language interfaces and fine-tuning models for specialized tasks. The consistent response times and coherent outputs suggest that the underlying architecture and implementation are sound.

On the other hand, the zero-element test case reveals that additional work is needed to ensure robustness across the full range of possible system conditions. Several mitigation strategies could address this limitation. First, the training dataset could be augmented with synthetic edge cases including empty or minimal systems to provide more balanced coverage of the state space [18]. Second, hybrid architectures could be implemented that combine retrieval-based lookup for certain

TABLE II
EVALUATION OF POWER SYSTEM CHATBOT ON OBJECT COUNT QUERIES

ID	Question	Expected	Predicted	EM	F1	Hall.	Time (s)
1	How many buses are there?	0 buses	37 buses	F	0.86	F	2.03
5	How many generators are in the system?	0 generators	45 generators	F	0.86	F	2.02
9	How many loads does the case have?	0 loads	27 loads	F	0.86	F	2.02
13	How many branches are there?	0 branches	89 branches (lines)	F	0.80	F	2.02
22	Total transmission lines please?	0 branches	89 branches (lines)	F	0.80	F	2.02
25	How many buses are there please?	0 buses	37 buses	F	0.86	F	2.01
27	Total generators please?	0 generators	45 generators	F	0.86	F	2.02
30	Count of loads please?	0 loads	27 loads	F	0.86	F	2.02
32	How many loads does the case have please?	0 loads	27 loads	F	0.86	F	2.02

query types with generative responses for others, ensuring that definitive questions receive definitive answers.

V. CONCLUSIONS AND FUTURE WORK

This work demonstrates a framework for developing domain-specific AI applications in power systems analysis. Automated conversion of simulation files to structured databases, large-scale question-answer dataset generation, and integration of retrieval-augmented generation with parameter-efficient fine-tuning can create functional natural language interfaces for specialized engineering tools.

The framework successfully addresses key challenges in making power systems knowledge accessible, including the automated extraction of structured data from proprietary simulation formats, the generation of training data grounded in actual system characteristics, and the adaptation of general-purpose language models to domain-specific tasks while maintaining computational efficiency.

Future work will pursue several promising directions. First, semantic similarity clustering of generated question-answer pairs could be implemented to identify redundancies and gaps in coverage, enabling more efficient and comprehensive training datasets. Second, the question template library could be expanded using advanced language models to generate diverse, natural-sounding questions that cover edge cases and complex analytical scenarios.

Third, time-series data could be incorporated, potentially including weather patterns, load profiles, and renewable generation forecasts, to enable dynamic simulation queries spanning temporal dimensions. Fourth, multi-modal outputs combining graphical visualizations with textual explanations could be explored, providing richer, more intuitive interfaces for system analysis and planning tasks.

Finally, advanced prompting strategies and few-shot learning techniques could be investigated to reduce the need for extensive fine-tuning while maintaining high accuracy [19]. These approaches may enable rapid adaptation to new domains or case types with minimal additional training data.

The promising results from this initial framework suggest that the integration of simulation tools with modern AI techniques can fundamentally transform how engineers and decision-makers interact with complex systems, making specialized knowledge more accessible and supporting more intelligent, explainable decision support across critical infrastructure domains.

REFERENCES

[1] T. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[2] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," 2023.

[3] D. Zhang, X. Han, and C. Deng, "Review on the research and practice of deep learning and reinforcement learning in smart grids," *CSEE Journal of Power and Energy Systems*, vol. 4, no. 3, pp. 362–370, 2018.

[4] PowerWorld Corporation, "Powerworld simulator," https://www.powerworld.com, 2025.

[5] A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye, "Grid structural characteristics as validation criteria for synthetic networks," *IEEE Transactions on Power Systems*, vol. 32, no. 4, pp. 3258–3265, 2017.

[6] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.

[7] Y. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," 2023.

[8] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019, pp. 3982–3992.

[9] E. J. Hu *et al.*, "Lora: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022.

[10] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021, pp. 7319–7328.

[11] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," 2023.

[12] C. J. Date, *An Introduction to Database Systems*, 8th ed. Boston: Addison-Wesley, 2003.

[13] M. Owens and G. Allen, *The Definitive Guide to SQLite*. Berkeley, CA: Apress, 2006.

[14] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2383–2392.

[15] S. Mangrulkar, S. Deshpande, M. Baweja, U. Thakker, and S. Biderman, "Peft: State-of-the-art parameter-efficient fine-tuning methods," GitHub repository, 2022, https://github.com/huggingface/peft.

[16] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.

[17] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[18] A. Anaby-Tavor *et al.*, "Do not have enough data? deep learning to the rescue!" in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 5, 2020, pp. 7383–7390.

[19] J. Wei *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 24 824–24 837.