2024 IEEE International Performance, Computing, and Communications Conference (IPCCC) | 979-8-3503-6794-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/IPCCC59868.2024.10850042

Revolutionizing Wireless Modeling and Simulation with Network-Oriented LLMs

Jiewen Liu, Zhiyuan Peng, Dongkuan Xu, Yuchen Liu
North Carolina State University, USA

***Abstract*—The complexity of modern network infrastructure continues to grow, supporting a wide range of interconnected applications. Simulators are indispensable tools in this context, providing cost-effective and risk-free environments for experimentation and development. However, mastering these network simulators demands substantial domain-specific knowledge, even with comprehensive user manuals. Motivated by the capabilities of Large Language Models (LLMs), this paper introduces the network-oriented LLM as an intermediary between users and network simulators, aiming to offer an interactive, automated, and script-free simulation paradigm. Using the emerging Sionna simulator as a case study, we adapt the general-purpose LLM into a network-oriented LLM through joint parameter-efficient fine-tuning and retrieval-augmented generation, which then streamlines the complex simulation process through simple natural language queries. Comprehensive experiments with state-of-the-art LLMs demonstrate that the proposed method can effectively adapt LLMs for use with network simulators, significantly enhancing user-level operational efficiency and accessibility. The proposed pipeline can facilitate the broader development of various network-oriented LLMs, potentially automating a range of complex network tasks.**

I. Introduction

Existing cyber-infrastructures are increasingly complex to support a vast array of applications, ranging from mobile communication, brain-computer interaction, flying vehicles, extended reality (XR), and industrial Internet of Things (IoT). The rapid evolution of these network systems demands rigorous testing and optimization to ensure reliability, utility, and security. Simulators are essential tools in this process, offering cost-effective and risk-free environments for experimentation and development. They enable detailed analysis and benchmarking of new technologies, facilitate safe validation of network configurations without the need for physical deployment, and provide the scalability to model interconnected networks of various sizes and complexities.

To date, several advanced network simulators have been developed to facilitate the rapid evolution of next-generation wireless network technologies. For instance, [1] developed ns-3, an open-source discrete event network simulator renowned for its high-fidelity, script-based simulations. It supports various network protocols, making it particularly suitable for research and development that necessitates realistic packet-level behavior. [2] introduced OMNeT++, a simulator celebrated for its modular architecture and user-friendly graphical interface, which excels in visualizing and debugging simulations. Another significant research thrust is the ray-tracing-based simulation, such as Wireless Insite [3], a commercial three-dimensional ray-tracer designed for radio propagation modeling on a city scale. More recently, NVIDIA developed Sionna [4], a TensorFlow-based open-source library tailored for simulating the physical layer of wireless and optical communication systems. Sionna enables link-level, differentiable simulations through the seamless integration of neural networks, representing a significant advancement in the field. By leveraging its ray-tracing module and just-in-time computation, researchers and developers can push the boundaries of network technology, enabling more efficient and reliable communication system design.

However, despite the availability of extensive simulators, mastering them always requires significant domain-specific knowledge, even with the help of dedicated user manuals. For instance, most of these tools, with limited GUI support, are known for their complexity and steep learning curve, making it challenging for new learners to set up and run customized simulations. Fortunately, the emergence of Large Language Models (LLMs) such as ChatGPT [5] and Llama [6] have revolutionized the field of natural language processing. With billions of parameters pre-trained on vast corpora, LLMs exhibit extraordinary abilities in understanding and generating human-like text and code snippets while simultaneously possessing domain-specific knowledge. This enables them to facilitate a wide range of applications such as content creation [7], medical diagnosis [8], and financial analysis [9]. Through in-context learning and parameter-efficient fine-tuning, LLMs can be adapted to master various domain-specific toolkits. Serving as an intermediate layer between humans and machines, LLMs are reshaping tool design and daily workflows through automation [10]. Inspired by this trend, several parallel works have emerged, leveraging LLMs in the realm of computer networks. Particularly, [11] proposes NetLLM for efficient adaptation of LLMs to network tasks, showing effectiveness in bitrate streaming and cluster job scheduling [12], while [13] leverages a Mixture of Experts (MoE) framework augmented with LLMs to efficiently analyze user objectives for diverse network optimization tasks. Additionally, [14] proposes an LLM-assisted end-to-end intelligent management framework that integrates semantic rule trees with LLMs for multi-scale diagnosis in dynamic heterogeneous networks. Meanwhile, a series of surveys and tutorials have been proposed [15]–[17], mainly describing the fundamentals, applications, challenges, and future directions in wireless networks.

Despite this promising potential, to the best of our knowledge, the integration of LLMs with network simulators for

wireless modeling remains unexplored. Motivated by this, we develop the first network-oriented LLM to master the simulator for end-to-end wireless optimization. To pioneer the pipeline for such an LLM-empowered simulation mechanism, we first create an instruction-following dataset containing question-answer pairs generated from the intricate tutorial documents. This dataset then serves as a basis for training and evaluating our network-oriented LLMs. Specifically, we adapt pre-trained LLMs from general-purpose to network-oriented using parameter-efficient fine-tuning (PEFT) and retrieval-augmented generation (RAG). For PEFT, we adopt low-rank adaptation [18] to fine-tune a pre-trained LLM through instruction-following training method [19]. The LLM is tuned to generate ground-truth answers given corresponding domain-specific questions, e.g., aiming to compress external knowledge from Sionna into the model. However, this approach often struggles with generalizing to unseen tasks and suffers from hallucination problems [20]. To address these issues, we employ RAG, which leverages LLMs' remarkable in-context learning capability [20]. RAG enhances user queries by retrieving the top-K relevant contextual chunks from the original tutorial documents. These augmented queries, along with the contextual chunks, are then fed into the LLM to produce more accurate and factual responses with fewer hallucination errors. With this overall design, users can now perform simulations through straightforward natural language prompting, reshaping the operation of current network simulators and automating complex workflows that previously demanded extensive manual intervention. This approach not only enhances operational efficiency but also *democratizes* the simulation process, making it *accessible* to a broader range of users, including those without specialized technical expertise. The contributions of this work are summarized as follows:

1) We present the first study that integrates LLMs with a network simulator. Using Sionna as a case study, we employ novel joint parameter-efficient fine-tuning and retrieval-augmented generation techniques to adapt general-purpose LLMs into network-oriented LLMs.
2) We develop a methodology for creating question-answer datasets regarding the simulator utility to facilitate the development of network-oriented LLMs. The generated data acquisition scripts can be readily modified to create datasets for other well-documented simulators.
3) Comprehensive experiments conducted on state-of-the-art LLMs indicate that model fine-tuning and advanced RAG techniques can effectively adapt LLMs for use with network simulators, significantly enhancing user-level operational efficiency. Example responses showcase that our network-oriented LLM can automate Sionna for ray-tracing analysis, radio map generation, channel impulse response calculation, etc.

II. Related Work

A. Network Simulation for Wireless Communication

Network simulation frameworks have been instrumental in advancing research and development in wireless communications. For instance, packet-level simulations focus on detailed protocol behavior and traffic patterns such as in ns-3 [21], [22] and GNS3 [23], while link-level simulations emphasize physical layer interactions and signal propagation characteristics as in Wireless Insite [3] and Sionna [4]. Notably, Sionna is a GPU-accelerated, open-source library built on the deep learning framework TensorFlow, designed for the fine-grained simulation of physical layers in wireless communication systems, and openly available to the community to drive 5G and 6G research [4]. However, mastering these network simulators always requires significant domain-specific knowledge and extensive scripting effort, making them less efficient and accessible to a broader range of users. Our work aims to bridge this gap with an automated LLM-enabled simulation paradigm.

B. Large Language Models

Language models, such as ChatGPT, have garnered global attention due to their user-friendly nature and ability to provide accurate information through plain text, revolutionizing the field of natural language processing (NLP). Pre-trained on extensive public datasets such as wikis, these models accumulate a broad knowledge base, demonstrating exceptional proficiency in a variety of applications, including logical reasoning [24] and code generation [25]. Their strong generalization capabilities enable them to apply learned knowledge to new, unseen tasks, effectively understanding and generating human-like text across diverse contexts. Inspired by these merits, an emerging trend involves deploying LLMs as foundational models for communication networks. However, to date, there is a domain mismatch between general-purpose language models and those specifically adapted for networking tasks. Recognizing this, researchers are working on adapting LLMs for various network applications [11], aiming to harness their full potential of these models in solving complex wireless network tasks, which is also the focus of this work.

III. Pipeline for LLM-Powered Network Modeling and Simulation

In this section, we introduce the design pipeline for incorporating LLMs into network simulations. Using the advanced simulator Sionna as a case study, we employ joint parameter-efficient fine-tuning and retrieval-augmented generation techniques to adapt general-purpose LLMs into network-oriented LLMs, automating the simulation process with interactive natural language prompting. As shown in Fig. 1, this framework consists mainly of three phases: data preparation, LLM fine-tuning, and the creation of the RAG system. Technically, the data generation engine creates question-answer pairs in natural language from the crawled tutorial documents. These question-answer pairs are then split into training and test data for fine-tuning and evaluating LLMs. In the instruction fine-tuning phase, a pre-trained LLM is employed, which takes a query as input and is tuned to generate the ground truth answer token-by-token in an autoregressive manner. In this way, the external knowledge of the simulator Sionna is lossily compressed into the fine-tuning LLM, which somehow can solve the problems

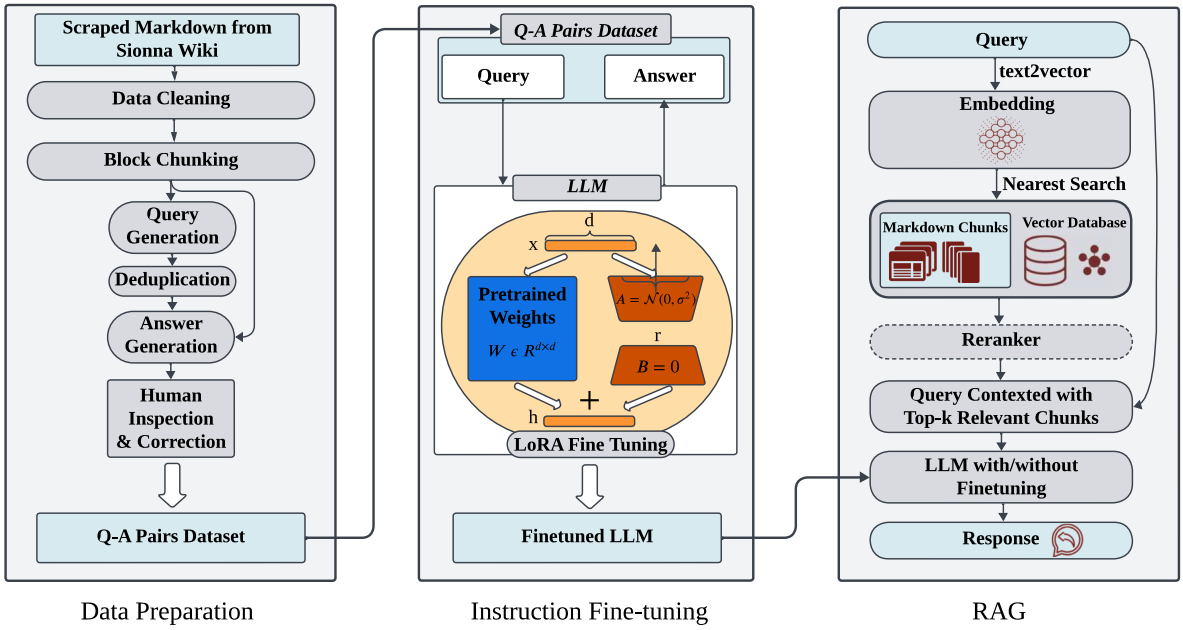


Fig. 1: Overview of the proposed workflow for developing network-oriented LLMs.

seen in training but is sensitive to the input queries and can hardly generalize to unseen cases, e.g., suffering from the well-known hallucination problem [20]. To this end, we then build a RAG system that first augments an input user query with the K most relevant contextual chunks retrieved from the original Sionna tutorial documents. Thus, both the query and the contextual chunks are fed into the fine-tuned LLM to facilitate a more comprehensive and factual response with reduced hallucination errors. More technical details are described as follows.

A. Data Acquisition

The first yet most critical step is data acquisition. The capability of instruction-following LLMs relies heavily on the quantity and quality of fine-tuning data. To enable instruction fine-tuning, data should be formatted as question-answer pairs. Conventionally, the question-answer pairs are manually curated, requiring extensive human effort and limiting dataset scalability. More importantly, creating question-answer pairs for specialized simulators like Sionna demands deep domain-specific knowledge, making the process even more challenging and time-consuming. Inspired by the remarkable progress made by the GPT4 series, we exploit these language models for automatic data acquisition.

Data Cleaning. As a case study, we choose the Sionna tutorial documents[1] as our foundation source to synthesize the required dataset. We first utilize a web scrapper to collect the necessary HTML content and convert it into markdown format. Empirically, using the markdown format yields better data generation quality compared to using plain text. Particularly, we remove embedded images, non-ASCII characters, HTML tags, tables of content, and hyperlinks in the documents, while equations are all converted into latex codes. The code scripts and their execution results embedded in tutorials are kept as the original.

Markdown Chunking. Before generating question-answer pairs, we split the Markdown files into several short *text chunks*. Unlike other prior RAG systems [26] that chunk plain texts, our system is built on markdown files, which consist of hierarchically organized texts in a rich format. To this end, we begin by parsing the extracted content into a hierarchical tree according to the heading levels specified in markdown files. Each node in the tree is associated with a heading and the corresponding texts within that section. Then, the markdown chunking method performs a pre-order traversal of the markdown tree following the *Root-Left-Right* policy. Next, we divide the text within each visited node into chunks that fall within a specified range of token counts. This ensures that each chunk is in an appropriate size for the question-answer generation. In practice, there remain a series of corner cases for the chunking process, which can be resolved by adding a series of hard-coded rules. Readers can refer to our chunking scripts[2] for the implementation details.

Automatic Generation of Question-Answer Pairs. For each chunk obtained in the previous step, we first prompt the LLM (e.g. gpt4-32k) to generate a list of questions that are relevant, comprehensive, and unique. The generated questions per chunk are post-processed again by prompting to reduce duplicated and off-topic ones. Then, we prompt the LLM to answer each generated question given the corresponding chunk. Fig. 2 provides the three prompt templates used for

[1]https://nvlabs.github.io/sionna/index.html

[2]https://github.com/ak-maker/sionna-LLMs/blob/main/RAG/code/preprocess/chunk.py

question generation, question deduplication, and answer generation. Since the LLM tends to be *lazy* about answering questions, e.g., generating incomplete codes or even simply responding with a reference to the original tutorial, we carefully curate the prompt as shown in Fig. 2 to force the LLM to generate complete answers, ensuring the quality of synthetic data.

B. Fine-tuning Network-Oriented LLMs

This stage aims to fine-tune a pre-trained decoder-based LLM $P_\Phi(\mathbf{y}|\mathbf{x})$ parameterized by Φ for instruction, where $\mathbf{x}$ is the input question and $\mathbf{y}$ is the generated answer. Given a corpus of question-answer pairs $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ extracted from the network simulation tutorials, the objective is to maximize the conditional probability distribution $P_\Phi(\mathbf{y}_i|\mathbf{x}_i)$ over $\mathcal{D}$, i.e.

$$\begin{aligned}\mathcal{L}_\mathcal{D} &= \max_\Phi \frac{1}{N}\sum_{i=1}^{N} \log P_\Phi(\mathbf{y}_i|\mathbf{x}_i) \\ &= \max_\Phi \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i} \log P_\Phi(y_{i,t}|\mathbf{x}_i, \mathbf{y}_{i,<t}),\end{aligned} \tag{1}$$

where $P_\Phi(\mathbf{y}_i|\mathbf{x}_i)$ is decomposed into $\prod_{t=1}^{T_i} P_\Phi(y_{i,t}|\mathbf{x}_i, y_i, < t)$ because of the iterative next-token prediction of LLMs. During the fine-tuning process, the parameters Φ are initialized from pre-trained weights Φ_0 and updated through stochastic gradient descent variants, e.g., AdamW [27]. The large-scale model parameters of LLMs make it challenging and even impractical to fully fine-tune all the parameters of Φ due to resource constraints and extensive computational cost. Therefore, we employ low-rank adaptation (LoRA) [18] for parameter-efficient fine-tuning [28].

In essence, pre-trained language models typically have full-rank weight matrices. LoRA assumes that the weight updates during adaptation possess a fairly low "intrinsic rank". Specifically, for a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, the update is constrained to a low-rank decomposition:

$$W_0 + \Delta W = W_0 + BA, \tag{2}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, with $r \ll \min(d, k)$. In the training stage, we can freeze the pre-trained weight W_0 without gradient updates. The trainable parameters are contained only by the matrices B and A. Both W_0 and $\Delta W = BA$ take the same input for multiplication, and their respective output vectors are summed element-wise. The modified forward pass is:

$$h = W_0 x + \Delta W x = W_0 x + BAx, \tag{3}$$

where B is sampled from a normal distribution and A is initialized to 0. In this way, the ΔW is set to zero and has no effect at the beginning of fine-tuning. LoRA can be applied to any subset of weight matrices in a neural network. In Transformer-based LLMs, this typically includes self-attention modules and the two feed-forward blocks.

C. Integration with Retrieval-Augmented Generation

The fine-tuning process compresses knowledge into LLMs, allowing them to answer questions encountered during fine-tuning. However, many unseen questions still require network knowledge the LLM has not yet acquired. In this regard, RAG can address this gap effectively by retrieving relevant contextual information from external sources, thereby enriching the model responses with precise and up-to-date knowledge. Specifically, we employ an efficient embedding model, text-embedding-3-small [29], to convert each markdown chunk into a compact fixed-dimensional embedding vector, which captures the semantic information of the markdown chunk. This creates a vector database linking each chunk with a vector, enabling efficient similarity searches among texts. For instance, when presented with a new network-domain question, we can convert it into an embedding and retrieve the K most relevant markdown chunks from the vector database. These retrieved markdown chunks are then added as the *context* to the original query and fed into our fine-tuned network-oriented LLM for answer generation.

To enhance the retrieval accuracy, we employ a re-ranker to calibrate the relevance score. The re-ranker adopts a two-tower architecture, wherein one tower encodes the query and the other encodes a more extensive set of candidate contents than initially required. It then subsequently reorders these contents based on their semantic relevance to the question, utilizing semantic relevance as opposed to the retriever's squared L_2 norm distance method [30], [31]. The utilization of such semantic relevance proves to be more effective in delivering high-quality responses from our network-oriented LLM, as validated in Sec. IV.

IV. Experiments and Discussion

In this section, we evaluate the effectiveness of our proposed network-oriented LLM for wireless modeling and simulation. We begin by evaluating the efficacy of our LLM fine-tuning. Following this, we assess the performance benefits provided by retrieval augmentation generation. Additionally, we compare the consistency of evaluations conducted by the LLM with those performed by human evaluators. Lastly, we demonstrate concrete use cases to illustrate the practical applications in network simulation.

A. Evaluation Setup

Dataset. We collect 48 markdown files from the Sionna tutorial documents and divide them into 727 chunks, primarily comprising texts and Python codes. These chunks are used for both question-answer generation and the build-up of the retrieval augmentation generation system. A soft constraint is applied to ensure most chunks fall within the length range of 25 to 800 tokens, as illustrated in Fig 3. Using the gpt4-32k model, 1,347 question-answer pairs are generated, which are then inspected and corrected by a human annotator. Among them, 150 pairs are randomly selected as the test set. To fine-tune the LLM, the remaining pairs are split into training and validation sets in a 9:1 ratio, resulting in 1,078 training pairs

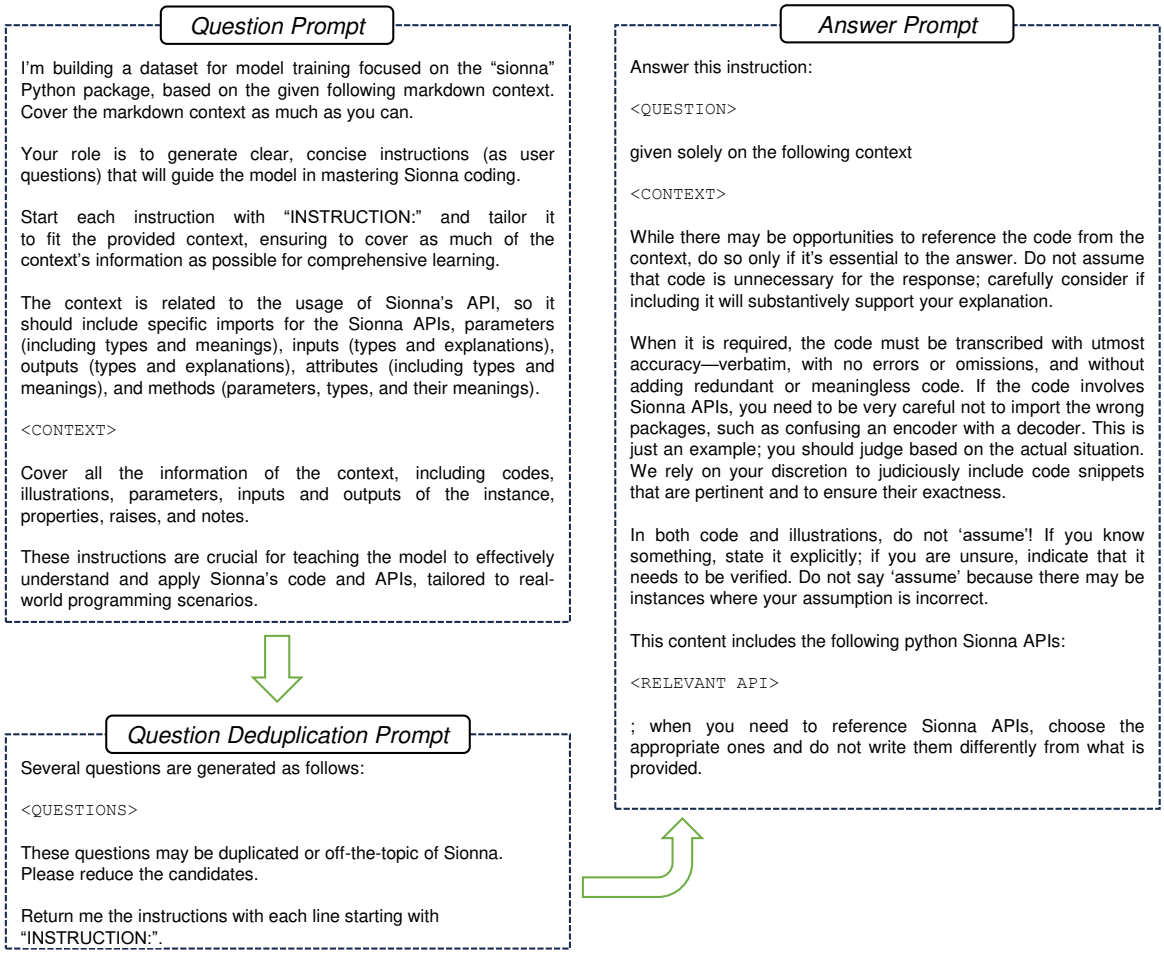


Fig. 2: Prompts used in generating question-answer pairs. Placeholders such as <CONTEXT> and <QUESTION> can be replaced with specific chunk content and questions. <RELEVANT API> encompasses a collection of programming language import statements.

and 119 validation pairs. Of these, 579 pairs contain purely textual answers, while 768 pairs include code segments.

Models. We use a series of LLMs from OpenAI for experiments, including GPT-3.5-turbo, GPT-3.5-turbo-16k, GPT-4-32k, GPT-4-1106-preview, and GPT-4-0125-preview. For simplicity, we denote them as gpt35, gpt35-16k, gpt4-32, gpt4-1106, gpt4-0125, respectively. We employ the gpt35 model for fine-tuning, using the Azure fine-tuning APIs [32]. The batch size is set to 32 across all experiments. However, we also experimented with varying batch sizes of 4, 8, 16, 24, and 32 to assess their impact. The text embeddings of the chunks are generated using *text-embedding-3-small* model [29], with a dimension of $1,536$. For the re-ranking process, we utilize *rerank-english-v3.0* from Cohere [33], a two-tower model, with one tower encoding a question and the other encoding a broader set of candidate contents than required. Such a re-ranker reorders these contents based on their relevance to the network-domain question. Following this methodology, the K most relevant content chunks are selected and appended to the original query.

Evaluation. Typically, evaluating the answer generated by LLMs is challenging. Recent works show that LLMs can act as evaluators who exhibit high agreement with humans [34] [35]. There is also potential in fine-tuning LLMs to act as evaluators in open-ended scenarios [36]. In this work, we use gpt4-32k as the evaluator due to its extensive context window. Given a specific question, the evaluator is prompted to score a generated answer according to a reference answer provided in the dataset, and the score range is limited to 1 to 9. To align the evaluation consistency with human preference, we design a prompt that divides the score range into three levels of correctness: 1–3 as *Limited Correctness*, 4–6 as *Partially Correct*, and 7–9 as *Substantial Correctness*. Inspired by the evaluation criteria in [37], we provide each level with a question-answer pair scored by a human. The prompt to the evaluator is also tuned to ensure score consistency with the human preference.

B. Fine-tuned Network-oriented LLMs

In this section, we start by evaluating the internal knowledge encoded in LLMs. In other words, we discard the last retrieval augmentation block shown in Fig. 1. The LLMs answer a user

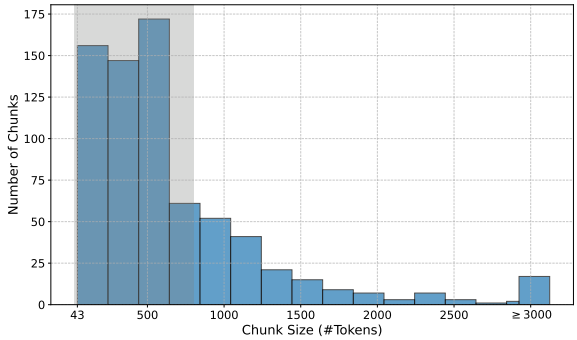


Fig. 3: Analysis of chunk size distribution: most chunks fall within the range of [25, 800].

query without any context and rely solely on their internal knowledge. The adopted evaluator is validated by *gpt4-32k* as described in later Section D.

Fig. 4(a) shows the average score of different LLMs from OpenAI, where *gpt35*, *gpt35-16k*, and *gpt4-32k* are the earliest GPT models released with training data collected up to September 2021. Since Sionna was first released in February 2022, these LLMs do not have any internal knowledge that is directly relevant to Sionna-based simulation and therefore, they exhibit poor performance scores as expected. In particular, *gpt4-32k* frequently rejects to answer questions due to the lack of network-domain knowledge, providing responses such as "*I'm sorry, I don't have information on that...*". While achieving the lowest score, the responses from *gpt4-32k* exhibit fewer hallucination errors, as it tends to avoid providing unreliable solutions. By contrast, *gpt35* and *gpt35-16k* tend to hallucinate, responding with incorrect answers to trick the evaluator for higher performance scores. *gpt4-1106* and *gpt4-0125*, on the other hand, have the training data up to April 2023 and December 2023, respectively. As such, they achieve noticeably better performance due to the potentially updated internal knowledge about Sionna.

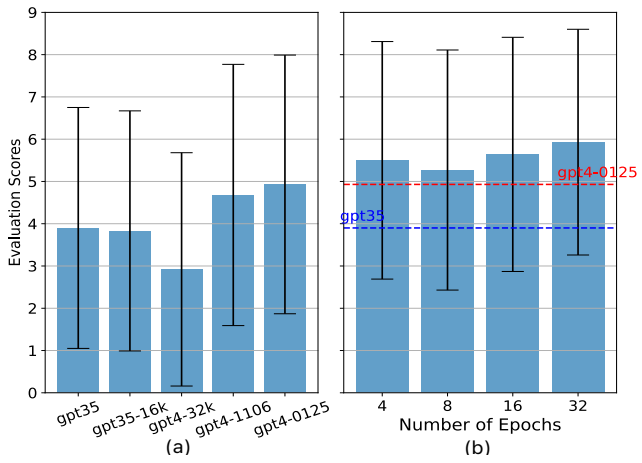


Fig. 4: Evaluation of LLMs for Sionna-based simulation. *(a)* the general-purpose OpenAI LLMs. *(b)* the fine-tuned network-oriented LLM with a different number of epochs.

This observation motivates us to update the network-domain knowledge of *gpt35*through the proposed parameter efficient fine-tuning as introduced in Sec. B. As shown in Fig. 4(b), the increase of training epochs gradually improves the mean score of answer quality and slightly reduces the standard deviation, showing our fine-tuning progressively updates task-relevant knowledge into the LLM. Notably, the fine-tuning provides a significant performance improvement over the original *gpt35* and even consistently outperforms the latest *gpt4-0125* by a large margin. This observation suggests that our fine-tuning method effectively adapts the general-purpose LLM into a network-oriented LLM.

C. Network-oriented LLMs with Retrieval Augmentation

Next, we analyze the effect of the RAG block introduced in Sec. C. Due to the limitations on input context size, we can only provide the *gpt35* series with the top-1 retrieved context. As shown in Fig. 5, the retrieval augmentation provides a consistent and significant performance improvement over all the LLMs. Such benefit of retrieval augmentation is orthogonal to the fine-tuning process, suggesting their *compatibility* in improving the model generation capability. Among the fine-tuned models, training with 8 epochs emerges as the best performer, compared to the 32-epoch model. It is important to note that the fine-tuning data does not include context information, so training with more epochs may lead to a slight mismatch during testing, resulting in unexpected performance degradation. Remarkably, the latest GPT-4 models, i.e. *gpt4-1106* and *gpt4-0125*, achieve the best performance due to their in-context learning capabilities. Besides, the proposed context re-ranking process brings further significant performance improvements. This suggests the relevance of retrieved contexts to the question plays a critical role in answer generation.

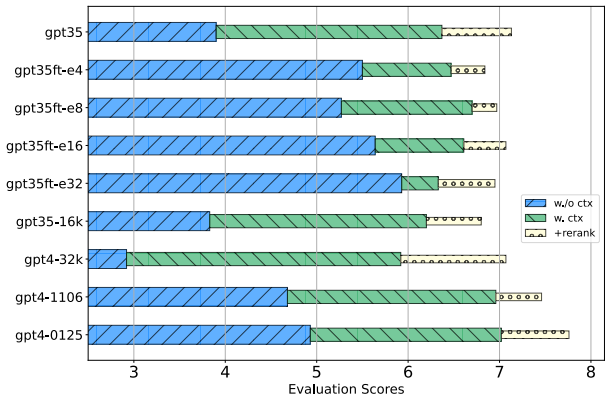


Fig. 5: Evaluation of network-oriented LLMs with retrieval augmentation. Only the most relevant chunk is used.

Inspired by this finding, we then explore adding more retrieved contexts into our network-oriented LLMs, aiming to maximize their in-context learning capabilities. As shown in Fig. 6, it is observed that adding more contexts can enhance the model performance, with the improvements saturated at the number of retrieved contexts k=3. Particularly, *gpt4-32k*

and *gpt35-16k* exhibit similar performance levels, whereas *gpt4-1106* and *gpt4-0125* consistently demonstrate a noticeable performance margin over the former two. These results underscore the effectiveness of integrating our RAG block into the network-oriented model with additional relevant contexts.

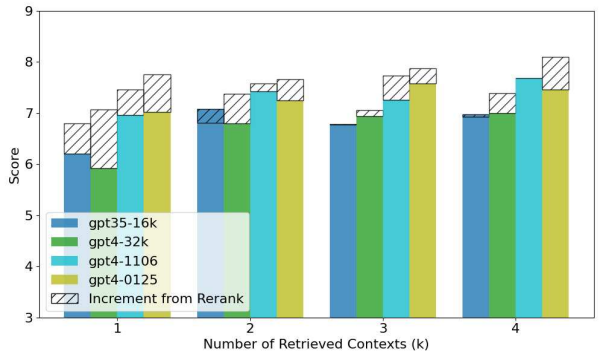


Fig. 6: RAG with a varying number of retrieved contexts.

D. Consistency between LLM-based and Human Evaluation

To validate the reliability of the adopted LLM-based evaluator, we analyze the score consistency between the LLM evaluation and the human evaluation. Specifically, we select 100 question-answer pairs from the test set and evaluate the results generated from *gpt35ft-e32* with the reference answers from the test set. Besides, a human expert in the network domain manually scores the answers. The human evaluation criteria vary by answer type, whether text or code. For code, the assessment focuses on two main criteria: 1) overall correctness, which examines whether the simulation code flow correctly addresses the question, and 2) adherence to relevant Sionna tutorials, including tool usage parameters and simulation methods. Text answers are directly scored based on keyword presence and relevance to the question.

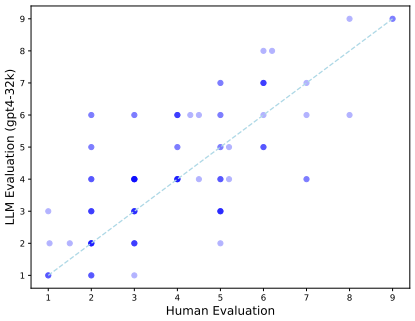


Fig. 7: Scatter plot of LLM-based evaluation and human evaluation. We compute the Pearson correlation coefficient with $r = 0.71$, suggesting a strong positive correlation between the two evaluators.

Fig. 7 presents a scatter plot depicting the relationship between evaluations conducted by LLMs and human experts. The discrete nature of the scores results in overlapping points, with the darkness of the points indicating the number of overlapping samples. The plot reveals a strong positive correlation between LLM and human evaluations, with some slight variance. The Pearson correlation coefficient is 0.71, indicating a strong positive correlation between the two evaluation methods.

E. User Cases for Wireless Modeling and Simulation

Lastly, we showcase the practical application of our network-oriented LLM for end-to-end simulations using straightforward natural language prompts. Our LLM provides step-by-step explanations through textual answers, code snippets, and figure plots. Fig. 8 shows the results generated by executing codes produced by our network-oriented LLM. This includes tasks such as network scenario creation, ray-tracing analysis, radio map generation, and channel impulse response calculation, which streamline the complex simulation process through simple user queries. More demo examples and resources can be found in our GitHub repository [38].

V. Conclusion

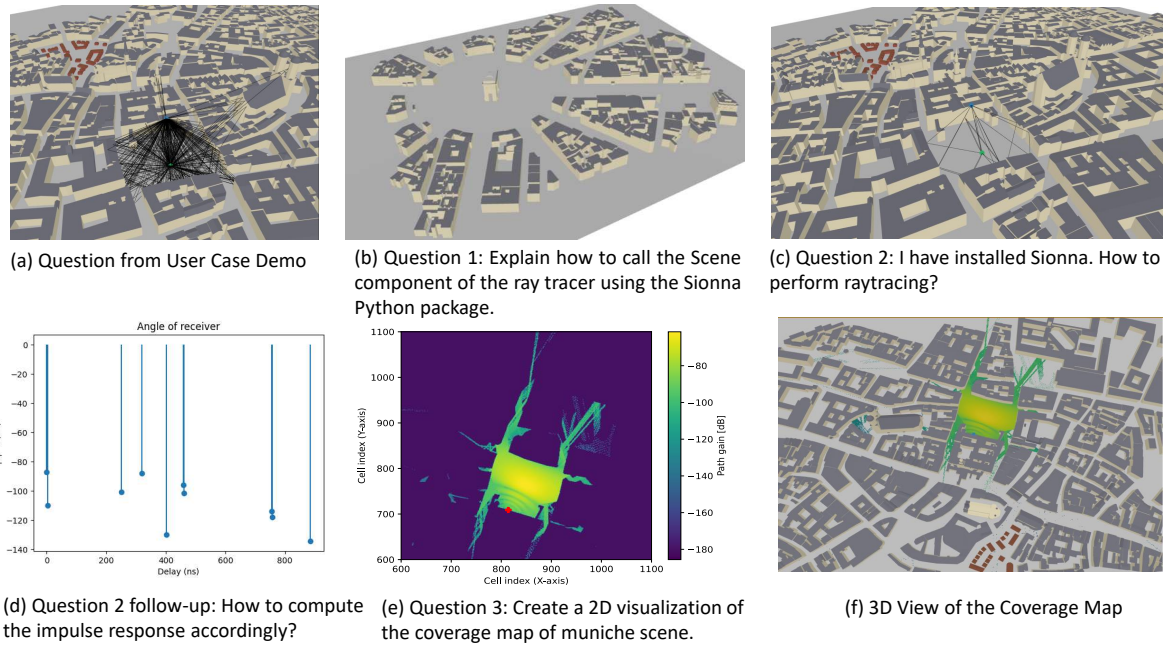
In this work, we investigated the integration of LLMs with the network simulator to streamline operations and reduce the learning curve associated with domain-specific simulation knowledge. We developed a specialized dataset for the Sionna simulator, fine-tuned the LLM to be a network simulation expert, and implemented a retrieval-augmented generation system tailored for executing the simulator. Comprehensive experiments assessed the models' proficiency in understanding and generalizing network-related knowledge and simulation code snippets. Our user case demonstration highlights how our approach effectively bridges the gap between plain text inputs and domain-specific knowledge, enabling an intuitive and accessible simulation experience. Future work involves supporting image inputs and multi-turn dialogues in the pipeline to enhance information accessibility and enable context-aware user interactions.

Acknowledgement

This research was supported by the National Science Foundation through Award CNS–2312138.

References

[1] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.
[2] A. Varga, "Omnet++," in *Modeling and tools for network simulation*. Springer, 2010, pp. 35–59.
[3] F. Remcom, "Wireless insite: 3d wireless prediction software," 2021.
[4] J. Hoydis, S. Cammerer, F. Ait Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, "Sionna: An open-source library for next-generation physical layer research," *arXiv preprint*, Mar. 2022.
[5] OpenAI, "Chatgpt," https://chat.openai.com/chat, 2024.
[6] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.
[7] T. Wang, J. Chen, Q. Jia, S. Wang, R. Fang, H. Wang, Z. Gao, C. Xie, C. Xu, J. Dai *et al.*, "Weaver: Foundation models for creative writing," *arXiv preprint arXiv:2401.17268*, 2024.
[8] J. Qian, Z. Jin, Q. Zhang, G. Cai, and B. Liu, "A liver cancer question-answering system based on next-generation intelligence and the large model med-palm 2," *International Journal of Computer Science and Information Technology*, vol. 2, no. 1, pp. 28–35, 2024.
[9] H. Yang, X.-Y. Liu, and C. D. Wang, "Fingpt: Open-source financial large language models," *arXiv preprint arXiv:2306.06031*, 2023.



(a) Question from User Case Demo

(b) Question 1: Explain how to call the Scene component of the ray tracer using the Sionna Python package.

(c) Question 2: I have installed Sionna. How to perform raytracing?

(d) Question 2 follow-up: How to compute the impulse response accordingly?

(e) Question 3: Create a 2D visualization of the coverage map of muniche scene.

(f) 3D View of the Coverage Map

Fig. 8: Simulation results from the network-oriented LLMs with Sionna executed by the RAG system.

[10] X. Liu, Z. Peng, X. Yi, X. Xie, L. Xiang, Y. Liu, and D. Xu, "Toolnet: Connecting large language models with massive tools via tool graph," *arXiv preprint arXiv:2403.00839*, 2024.

[11] N. Team, "Netllm: Adapting large language models for networking," *IEEE Communications Magazine*, 2024.

[12] D. Wu, X. Wang, Y. Qiao, Z. Wang, J. Jiang, S. Cui, and F. Wang, "Large language model adaptation for networking," *arXiv preprint arXiv:2402.02338*, 2024.

[13] H. Du, G. Liu, Y. Lin, D. Niyato, J. Kang, Z. Xiong, and D. I. Kim, "Mixture of experts for network optimization: A large language model-enabled approach," *arXiv preprint arXiv:2402.09756*, 2024.

[14] F. Tang, X. Wang, X. Yuan, L. Luo, M. Zhao, and N. Kato, "Large language model (llm) assisted end-to-end network health management based on multi-scale semanticization," *arXiv preprint arXiv:2406.08305*.

[15] Y. Huang, H. Du, X. Zhang, D. Niyato, J. Kang, Z. Xiong, S. Wang, and T. Huang, "Large language models for networking: Applications, enabling techniques, and challenges," *arXiv preprint arXiv:2311.17474*, 2023.

[16] Y. Du, S. C. Liew, K. Chen, and Y. Shao, "The power of large language models for wireless communication system development: A case study on fpga platforms," *arXiv preprint arXiv:2307.07319*, 2023.

[17] C. Liu, X. Xie, X. Zhang, and Y. Cui, "Large language models for networking: Workflow, advances and challenges," *arXiv preprint arXiv:2404.12901*, 2024.

[18] 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," *arXiv preprint arXiv:2106.09685*, 2021.

[19] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[21] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34.

[22] Y. Liu, S. K. Crisp, and D. M. Blough, "Performance study of statistical and deterministic channel models for mmwave wi-fi networks in ns-3," in *Proceedings of the 2021 Workshop on ns-3*, 2021, pp. 33–40.

[23] J. C. Neumann, *The book of GNS3: build virtual network labs using Cisco, Juniper, and more*. No Starch Press, 2015.

[24] Y. Zhang, S. Mao, T. Ge, X. Wang, A. de Wynter, Y. Xia, W. Wu, T. Song, M. Lan, and F. Wei, "Llm as a mastermind: A survey of strategic reasoning with large language models," *arXiv preprint arXiv:2404.01230*, 2024.

[25] R. A. Poldrack, T. Lu, and G. Beguš, "Ai-assisted coding: Experiments with gpt-4," *arXiv preprint arXiv:2304.13187*, 2023.

[26] Y. Song, R. Yan, X. Li, D. Zhao, and M. Zhang, "Two are better than one: An ensemble of retrieval-and generation-based dialog systems," *arXiv preprint arXiv:1610.07149*, 2016.

[27] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[28] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," *arXiv preprint arXiv:2101.00190*, 2021.

[29] OpenAI, "New embedding models and api updates," 2024, accessed: 2024-07-01.

[30] Cohere, "Rerank," https://docs.cohere.com/docs/rerank-2, 2024, accessed: 2024-07-02.

[31] Chroma, "Usage guide," https://docs.trychroma.com/guides, 2024, accessed: 2024-07-02.

[32] Microsoft, "Azure openai gpt-3.5 turbo fine-tuning tutorial," 2024, accessed: 2024-07-01.

[33] Cohere, "The leading enterprise ai platform," https://cohere.com/, 2024, accessed: 2024-07-02.

[34] C.-H. Chiang and H.-y. Lee, "Can large language models be an alternative to human evaluations?" *arXiv preprint arXiv:2305.01937*, 2023.

[35] Y. Dubois, C. X. Li, R. Taori, T. Zhang, I. Gulrajani, J. Ba, C. Guestrin, P. S. Liang, and T. B. Hashimoto, "Alpacafarm: A simulation framework for methods that learn from human feedback," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[36] L. Zhu, X. Wang, and X. Wang, "Judgelm: Fine-tuned large language models are scalable judges," *arXiv preprint arXiv:2310.17631*, 2023.

[37] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing *et al.*, "Judging llm-as-a-judge with mt-bench and chatbot arena," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[38] J. Liu, "sionna-llms," https://github.com/ak-maker/sionna-LLMs, 2024, accessed: 2024-07-04.