slAIces: an LLM Chatbot for Simplifying Experiments with the SLICES-RI

Dimitris Kefalas[⋆†], Sokratis Christakis[⋆†], Serge Fdida[⋆],
Nikos Makris[†‡], Ilias Syrigos[†‡], Virgilios Passas[†‡] and Thanasis Korakis[†‡]
[⋆]*Laboratoire LiP6/CNRS, Sorbonne University, Paris, France*
[†]*Dept. of Electrical and Computer Engineering, University of Thessaly, Greece*
[‡]*Centre for Research and Technology Hellas, CERTH, Greece*
Email: dkefalas@uth.gr, schristakis@uth.gr, serge.fdida@lip6.fr,
nimakris@uth.gr, ilsirigo@uth.gr, vipassas@uth.gr, korakis@uth.gr

***Abstract*—SLICES-RI is the outcome of several years of evolution of the concept of a networking test platform, that has recently transformed into a scientific instrument for the assessment of new research challenges in the Digital Infrastructures domain. Making large-scale scientific instruments easily accessible has always been challenging, often hindered by the complexity of the underlying platforms, the size and diversity of resources and the low-level settings that experimenters need to configure. Emerging technologies such as artificial intelligence (AI) have ignited a wave of innovation, inspiring companies, professionals, and researchers to incorporate chatbot assistants into their projects. These assistants revolutionize user experience by simplifying access to information and resources, significantly increasing productivity and presenting a familiar interface. In this work, we introduce slAIces, an LLM-based chatbot specifically designed to ease access to the SLICES-RI. slAIces utilizes the Generative Pre-trained Transformer (GPT) GPT-4 model to create a sophisticated Retrieval Augmented Generation (RAG) system. This system provides external knowledge by leveraging a properly preprocessed dataset, which includes documentation from SLICES-RI and all integrated testbeds. The contribution of this study lies in its methodology and recommendations for enriching the information on the test platform to enhance the quality of the service provided. We demonstrate that this chatbot significantly reduces the learning curve for new experimenters to become acquainted with the infrastructure. It exposes the appropriate level of abstraction, enabling experimenters to conduct complex experiments mobilizing the extensive and diverse resources available within a large-scale infrastructure like SLICES-RI.**

***Index Terms*—LLM, RAG, chatbot, SLICES, Research Infrastructure, GPT**

I. Introduction

Research Infrastructures (RIs) have emerged as key components for validating research outcomes in domain sciences. Such instruments need to keep up with the most recent advances in technology, offering bleeding-edge resources to researchers, and integrating the most advanced features that accelerate the materialization of new ideas to research outcomes. As such, SLICES [1], [2] is the first RI defined as a scientific instrument in the Digital Infrastructures (DI) domain, able to offer cutting-edge resources for experimentation to researchers of the field, while providing them with structured experiment lifecycle management from the design phase, to the deployment, evaluation, archiving and post-experiment analysis. The resources offered by SLICES are defined based on thought experiments, spanning a wide range of programmable infrastructure, from Open-RAN and beyond 5G networks to edge and High-Performance Computing (HPC) and IoT.

However, retrieving documentation needed for bootstrapping the experimentation process, as well as identifying the appropriate resources in such complex environments can be a very tedious task, especially for new users of the RI that are not yet familiar with the specific methodologies that need to be followed in order to get started with the experimentation process. To this aim, tools that increase the user-friendly interface for on-boarding users with the facilities are needed to increase the attractiveness and overall usability of the RI. Generative AI and Large Language Models (LLMs) have emerged as tools that can simplify to a large extent the creation of new human interpretable content based on existing knowledge that they have access to. Based on their training inputs, such models are able to create content by combining knowledge from multiple sources, including even automatic code generation, that can be used in the context of user support for RIs. In this work, we present slAIces, an LLM-based chatbot for lowering the barrier for new users experimenting with the SLICES-RI. slAIces uses the well-established Generating Pre-trained Transformer (GPT) 4 model for generating output for accessing the infrastructure, including selecting the most appropriate nodes depending on the type of experiment that the users need to conduct, while receives as input the documentation of each SLICES-RI testbed (incl. hardware capabilities, supported types of experiments, etc.). This manuscript presents the design choices, lessons learned and recommendations from the development and the model training process, and a framework evaluation based on the inputs provided from the online documentation sites of the different SLICES-RI nodes.

The rest of the paper is organized as follows: Section II presents our related work and different models that can be used for generating content. Section III presents our overall architecture and the selected LLM configuration for the slAIces

The research leading to these results has received funding from the European Horizon 2020 Programme for research, technological development and demonstration under Grant Agreement Number No 101008468 (H2020 SLICES-SC). The European Union and its agencies are not liable or otherwise responsible for the contents of this document; its content reflects the view of its authors only.

ISBN 978-3-903176-63-8 © 2024 IFIP

agent. In Section IV, we evaluate our agent under different training inputs, highlighting the importance of the choices made during the design and development process. Finally, in Section V we conclude and present some future directions.

II. Motivation & Related Work

A researcher, with no familiarity with the SLICES infrastructure, can benefit from the capabilities provided by the slAIces chatbot. Indeed, it can significantly reduce the time it takes to learn the experimental platform and help the experimenter unlock the full potential of SLICES-RI much more rapidly than traditional methods would allow.

When designing an LLM-based chatbot, it is important to consider how the LLM model will acquire special-purpose knowledge. Authors of paper [3], present insights on how LLMs can be used to integrate and apply domain-specific knowledge with the usage of in-context learning (ICL), providing the techniques, and challenges of ICL. One widespread way of integrating an external source of knowledge into LLM, based on ICL and structuring the retrieval information part, is through Retrieval-Augmented Generation (RAG) systems. Paper [4] provides a brief analysis of the architecture of the different components and the advancements of RAG systems, analyzing different sophisticated models such as Naive RAG, Advanced RAG, and Modular RAG. It also proposes methods for evaluating a RAG system, many of which were utilized to evaluate our proposed system in section IV. Paper [5], explores a different way to construct domain-specific LLMs via Fine Tuning (FT). The authors implement a comprehensive approach for the FT process, analyzing various aspects such as dataset selection, preprocessing, model choice, particularly focusing building LLMs for the financial field.

Authors of [6] explore applying an LLM model using the domain-specific knowledge of surface engineering, having as a dataset a list of scientific papers of the field, using the indexing method. This work highlights the importance of integrating advanced AI tools with expert knowledge, to enhance accuracy and relevance in a specialized field (such as the surface-engineering) oriented queries. Exploring the integration of LLMs into specialized domains, it is worth mentioning the work [7], which provides LLM-powered agent assistance into the video editing workflow, aiming to simplify complex editing tasks such as automatically generating descriptions for footage, enabling the LLM to execute editing tasks based on user objectives and others. The survey in [8] explores the use of LLMs in developing autonomous agents, noting a departure from previous research focused on agents with limited knowledge in isolated environments. It presents a detailed examination of the architecture, diverse applications, methodologies for assessment and challenges of LLM-based autonomous agents, thus providing very important details on the design and implementation of slAIces. The paper [9] delves into the evolution and challenges faced by Virtual Intelligent Assistants (VIAs) such as Amazon Alexa and Google Assistant, which have gained prevalence in recent years. It explores new research directions focused on understanding multi-modal contexts, and personalizing assistant services to provide insights for researchers and practitioners while emphasizing the pivotal role of LLMs in enhancing these strategies.

In this work, we pioneer the development of an advanced RAG system using the special purpose data sourced from the SLICES RI documentation, enriched with a strategic usage of GPT4 and combining both ICL and FT methods. This system aims to provide an improved user experience for experimenters of the SLICES RI infrastructure to reduce the learning curve for new users, increase the research productivity and assisting in experimental design and planning.

III. Architecture and LLM configuration

To design SlAIces, we employed a structured framework development strategy, which consists of four key stages: data preparation, designing and implementing a RAG system, LLM fine-tuning, and user querying via an appropriate environment. The overall integrated architecture can be observed in Figure 1. The initial stage involves acquiring SLICES data from diverse websites using the web scraper described in subsection III-A. Subsequently, raw data is appropriately enhanced by enriching both the textual content and the code content. This enhancement process involves leveraging the online GPT-4 model, which is manually utilized to upload our data for further enrichment. In addition, our domain expertise is leveraged to supplement this process. The RAG system consists of a pipeline of different functions that are responsible: 1) for breaking and storing the data in suitable data structures (indexing), 2) retrieving the most similar information to the user query from the data (retrieval), 3) constructing a prompt that contains the retrieved information as long as the user query (prompting), 4) using an LLM to get the response of the user query (LLM-Output). Following these, we proceed with deploying the chatbot service onto one of our SLICES servers. In order to provide a user-friendly interface, we have established a link between the SLICES server (housing the deployed chatbot) and a Discord server. This integration allows users to interact with the chatbot directly through the Discord server, simplifying the process and providing a more intuitive interface for the end user. The server is also integrated to the online portal, to provide a conversational experience to the users that want to conduct experiments within the SLICES infrastructure. The basic concepts and functionalities of our system are further explained in the subsections below.

A. Data acquisition

The data that will feed our model are sourced primarily from the official website of SLICES-RI [10] as well as the links within its contents that point to educational and training material, such as tutorials for students and researchers to become familiar with various frameworks and technologies that each integrated testbed of SLICES-RI is equipped.

To be able to automatically retrieve the required information from the aforementioned documentation websites, a web scraper was developed, designed to operate in two stages. In the first stage, all the links and URLs inside the documentation

pages are tracked recursively from the root website, with the help of the *urllib* library. After gathering all the referral URLs on the visited websites, duplicate URLs are removed to create a unique set of URLs. Subsequently, a manual review and evaluation of the content of the web pages is performed, to evaluate its relevance and value to be included in the dataset. In the second stage of the web scrapper, the web page data is retrieved and converted from HTML format to text through the *Beautiful Soup* library. Given the variation in website designs among the SLICES-RI testbeds, the code was adapted according to each website-specific design to extract the information correctly. At this stage, the data from each website is stored in distinct files, allowing a modular approach to removing/adding new information to the model.

B. Data augmentation

This step is mandatory to make our system efficient. Although the retrieved data from the documentation sites are interpretable by humans, they can be less beneficial for LLMs. LLMs, in contrast to humans, struggle to understand complexity and ambiguity since they lack real-world understanding.

The learning process for humans versus the one for LLMs significantly differs, especially in tutorials. A human can approach tutorials already having a basic understanding of the technology and the ontology of the various entities and components used in them. Foundational knowledge makes it easier for humans to make new connections among the different parts of information as long as they can easily grasp new concepts. Tutorials mainly point to hands-on experience, not providing depth in theory. In addition, the reader of the tutorials can refer to other sources besides the tutorial and its own references. However, none of these applies to LLMs, since LLMs lack ontological understanding. They realize text based on patterns observed using statistical correlations in their training data and not using relationships to convey meaning. LLMs also suffer from Referencing Limitations since their knowledge is restricted only to their training and In-context Learning (ICL) dataset, without external post-training resources.

Our raw dataset, provided by the documentations, presents gaps and deficiencies for appropriate LLM training, with respect to clarity, depth, or specificity of different information. To address such vulnerabilities, we enriched our dataset with comprehensive vocabulary as well as with an extensive and precise description of the theory (e.g. component's architecture, terminology, etc.). We employed the popular ChatGPT AI system [11] based on GPT4 model [12] to perform this enhancement. Following that, we restructured the generated text, into smaller paragraphs with titles emphasizing the context of their body. We validated rigorously, and thoroughly checked the data enhancement to ensure that we did not add vague information and pollute our data with noise, since ChatGPT can make mistakes.

C. In-context learning & Fine Tuning

A diverse amount of publicly available internet text data, such as websites, Wikipedia, Common Crawl, articles, as well as scientific papers, have been used to train GPT. This extensive training leads LLMs to identify patterns observed using statistical correlations, and identify grammar, syntax, and semantics, achieving a significant amount of general knowledge. Although GPT has tremendous performance in general-purpose knowledge, it looks very vulnerable to specific domains, such as providing assistance to SLICES RI infrastructure, where domain-specific knowledge and contextual understanding are crucial. For an LLM model to be updated with special-purpose data, there are two popular methods: in-context learning via a RAG system and fine-tuning.

1) In-context learning via a RAG system: The first approach, in-context learning, feeds the LLM with special-purpose data and by providing extra information in the prompt, without any parameter updates or fine-tuning. This method utilizes a RAG system to insert context into the input prompt, allowing the LLM to generate responses by leveraging its reasoning capabilities. Our RAG implementation follows the Naive RAG architecture as described in [4] and has three necessary axes: Data Ingesting, data indexing and data retrieving.

Data ingestion refers to how the contents of files in different formats (CSV, JSON, XML, PDFs, etc.) can be converted into data structures. Subsequently, Data Indexing is responsible for organizing, storing, and efficiently accessing the converted data. Several proposed index schemes can be used in different use-case scenarios. A node in the data Indexing world, represents a chunk of source data from a document, regardless of its type. There are multiple index schemes available, such as List Index, Vector Store Index, Tree Index and the keyword index. The data retriever is responsible to identify the most similar data nodes of the indexed dataset to include its content to the prompt. Our system utilized the open-source library Llama [13] to manage the data ingestion, indexing and retrieving procedures of slAIces implementation.

2) Fine Tuning (FT): Fine-tuning (FT) is another strategy to integrate specific-domain knowledge into an LLM than ICL. The strategy mainly focuses on training a pre-trained model using domain-specific datasets to incorporate data from specialized fields. Using a pre-trained model offers considerable advantages compared to an LLM that is exclusively trained with our private data. One of the most important ones is efficiency, since training the model from scratch requires significant computational resources, time, and an enormous dataset to learn language understanding. FT changes the model's weight through additional training, improving performance on domain-specific tasks. During the FT process, great attention was given to the parameters that will be modified so that they maintain the performance of our model and not to become overfit.

The strategy of FT is not as straightforward as that of In-context learning. Since the FT process is designed to adjust the model's parameters using examples of input-output pairs, we have to change the format in which the data are fed to the model. Instead of giving the information in raw text as in Section III-C1, all information intended for training should be converted into a query-answer format to be used as a prompt.

Following this, our dataset should be structured in a prompt-completion format as described in the Fine-tuning Guide [14]. We decided to fine tune the model with a portion of the dataset that LLM had difficulty responding when using only ICL. OpenAI does not support fine-tuning for GPT-4 yet, since it is at an experimental stage at the time this manuscript is written. For this work, FT used GPT-3.5-turbo-0125 limiting the model's capabilities compared it to the GPT4.

ICL and FT are two complementary techniques that can enhance a model's capabilities on distinct levels without being mutually exclusive. Our chatbot assistant is of considerable complexity, primarily because it must deliver not just theoretical knowledge but also detailed instructions and advanced information across a multitude of configurations, console outputs, commands, architectural concepts and many more. To deal with this issue, it is critical to integrate both FT and ICL into our development process. With ICL we are capable of providing the model with all the information, including even the slightest detail. On the other hand, FT is inevitable, since it helps the model to deeply understand our specific dataset, enabling it to interpret the meanings of different data types and follow advanced procedures.

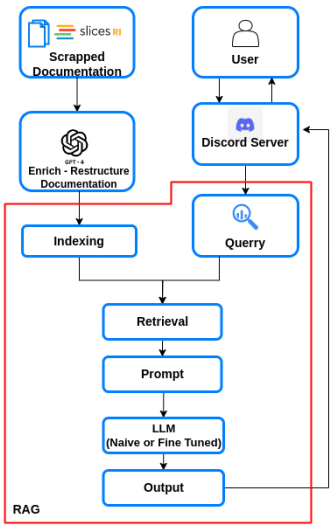


Fig. 1: System Architecture

IV. Evaluation

In Section IV-A, we describe our evaluation methodology between the generated dataset after the enrichment process (see Section III-B), and the raw one. Then, in Section IV-B we examine the system performance by leveraging multiple choice and open-ended questions using the original and the enhanced data as long as the fine-tuned LLM (Section IV-C).

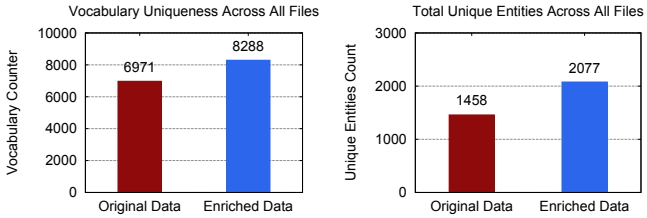
A. Datasets

To compare the enriched dataset against the raw one, we followed the popular dataset evaluation method described in [15], which analyzes critical LLM dataset characteristics such as data size, comprehensiveness, and how easily we can retrieve data from it.

1) Data size & Vocabulary uniqueness: As already mentioned, LLMs are based on identifying patterns observed using statistical correlations. The enhanced dataset is obviously larger than the raw data. Although, presenting only its size would not support the efficiency of our proposed dataset. That is why we focused on the vocabulary uniqueness, which defends that the diversity and the richness of the dataset not only offer a large quantity of text to the indexing mechanism of RAG, but also exposes it to a wide variety of words, phrases, and linguistic structures. This plays a crucial role, especially in the ICL and the way the RAG system will provide the necessary information to the model query.

The enriched data size is approximately 20% bigger than the raw one. Figure 2a, illustrates that the enriched data have 18% more uniqueness regarding the vocabulary, implying a highly efficient enrichment process without leading to an overload dataset, since the additional text is just a lexical restructuring of meaning and content of the raw data.

2) Data comprehensiveness: This lexical restructuring, alongside with the necessary explanation provided to some contexts of the original dataset (such as a more brief explanation of programming commands, more detailed technology architectural analysis, and theory), has led to a clearer ontological existence of entities in the enhanced dataset. Such an example of an entity could be the 5G components, different kind of testbed equipment and others. Entities have a crucial role in a dataset's comprehensiveness since they reflect real-world complexity into it. To validate the enrichment process in concern to a more structured and well-written enhancement, we utilized the biggest pre-trained model of SpaCy library, en_core_web_lg [16], which is a popular large English model designed for general-purpose Natural Language Processing (NLP) tasks, to identify entities.



(a) Overall Vocabulary Uniqueness Across All Files (b) Total Unique Entities Across All Files

Fig. 2: Data Comparison: Vocabuary & Entities Uniqueness

Following the results of the vocabulary analysis, we continued the evaluation of our enhanced dataset to its entity dimension, comparing it to the original one. As unique entities, we identify each distinct entity that the model was able to recognize in our datasets, counting each entity only once, regardless of how many times it appears across the data. This ensures that even if an entity is observed multiple times, it is included as a single entry in the final set of identified entities. Figure 2b, illustrates a 42.5% increase in the total unique entities of the enriched data identified by the entity model. It is worth mentioning that the model fed with the original data, could not understand some key components

of SLICES tutorials, such as the F1 interface, which is an important interface in the distributed 5G RAN architecture. These results indicate that the descriptive power and utility of the enhancement lead to an ML model being able to identify domain-specific terms and concepts.

3) Retriever Analysis: To assess our RAG implementation, we need to find suitable metrics, such as the similarity metric, that describe the process of finding the index's nodes that match the user queries. The proposed RAG system, implemented with the usage of the Llama library is based on the following strategy: Initially, the retriever is responsible for fetching the most relevant context from the available index nodes based on the query's degree of similarity and the node's content. The topology of the nodes could be one of those described in III-C1. Following that, the NodePostProcessor accepts a set of retrieved nodes as input and then applies filtering, augmentation, and replacement transformations. In our implementation, we used the Vector Store Indexing topology and the NodePostProcessor, to perform filtering based on the similarity metric. We set as a lower limit the value at 0.8, to support the model being fed with a lot of data with relevant content when we have a high degree of similarity, and with less when data's content is not so relevant. Finally, there is the response synthesizer, which takes the transformed set of nodes from the NodePostProcessor and, based on their content, forms the information that our system will provide to LLM (such as GPT-4 in our case) to generate the response.

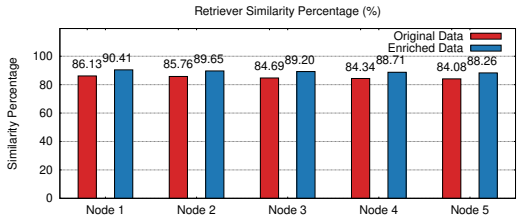


Fig. 3: Retriever Similarity Percentage

We used the retrieving process to evaluate the impact of our enhanced dataset over the raw one. As already mentioned, the retriever is able to identify nodes of the indexed data with similar content to the user query. Figure 3 illustrates the average similarity scores from the ith nodes obtained by the retriever for the open-ended questions described in Section IV-B. As depicted, the retriever in the case of enhanced data can recognize a higher similarity degree of the top_k nodes (5 in our case), having an average of 5% higher similarity over the raw data.

B. Performance Evaluation - In-context Learning

To further assess the chatbot's performance before and after data augmentation, we have also generated 25 multiple-choice questions and open-ended questions from our datasets, which can be found in [17]. It is important to mention that each question was formulated across a diverse range of data sources and was structured to have answers that had small deviations from the correct one, in order to increase the level of difficulty. Subsequently, the questions were first presented

TABLE I: Question Responses per Model & Dataset

Question	**Which nodes of the NITOS testbed should I use to deploy the 5G RAN using USRPs according to 5G Blueprint?**
Original Data with GPT-4 Model	The given context does not provide specific details about the nodes of NITOS that can deploy 5G RAN using USRPs according to the 5G Blueprint. Detailed information regarding this would be required to provide a precise answer.
Enhanced Data with GPT-4 Model	The NITOS testbed nodes that are capable of deploying the 5G RAN using USRPs, according to the 5G Blueprint, are: node055, node056, node059, node060, node067, node068, node071, node072. These nodes are equipped with USRP B210 models, which are connected via USB 3.0. The USRP B210 devices are particularly significant for 5G research due to their wide frequency range and high bandwidth capabilities, enabling researchers to simulate and analyze various aspects of 5G networks.
Enhanced Data - GPT-3.5	To deploy the 5G RAN using USRPs according to the 5G Blueprint in the NITOS testbed, you should utilize the nodes equipped with USRP N210 and USRP B210 devices. Here are the specific nodes that you should use for this purpose: USRP N210 Nodes: node052, node064, node080 USRP B210 Nodes (Connected via USB 3.0): node055, node056, node059, node060, node067, node068, node071, node072.
Enhanced Data - Fine-Tuned GPT-3.5	The NITOS testbed encompasses specific nodes equipped to facilitate the deployment of 5G RAN experiments utilizing USRP devices. According to the 5G Blueprint, the following NITOS nodes are suitable for this purpose: node055, node056, node059, node060, node067, node068, node071, node072. These nodes are instrumental in realizing advanced 5G RAN scenarios, enabling researchers to leverage the versatile capabilities of USRP B210 models for comprehensive and cutting-edge experiments.

to the RAG system with the original dataset, and then to the same model indexed with the enhanced dataset. The RAG system was prompted the same way for both the raw and the enhanced dataset, outlining the role of RAG as a chatbot assistant, as well as generating responses based only on the information provided to it. As depicted in Figure 4, the correctly-answered questions percentage by the model with the original-data was 76%, in contrast to the enhanced-data model that achieved 96%. This difference suggests a huge performance improvement, as it underscores the importance of data augmentation in enhancing the chatbot's understanding and response accuracy.

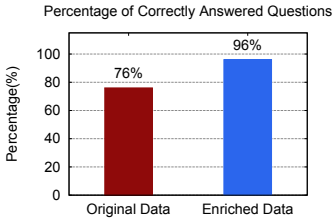


Fig. 4: Percentage of Correctly Answered Questions

Since there is no deterministic way to evaluate the validity of an open-ended response from LLMs formed from multiple data sources, we adopted the human validation method. Using our domain expertise and dataset familiarity, we checked the generated responses in terms of validity, closeness, clarity and descriptiveness for both datasets (original & enriched).

Regarding the open-ended answers, the model that uses the enhanced data provided responses with more precision and included more information compared to the model using the raw dataset. When it came to modifying code and configuration files, both models achieved the expected outcomes, with the responses from the RAG system using the enriched data being more solid, especially in the reasoning part. For example, as

depicted in Table I, when asking the same question (*Which nodes of the NITOS [18] testbed should I use to deploy the 5G RAN using USRPs according to 5G Blueprint [19]?*) to the GPT-4 model, the chatbot did not respond in the case of raw data, in contrast to the enhanced one, which answered correctly. At this point we note that the same information was present in both datasets, but the GPT-4 model could not successfully identify or interpret it to produce good responses.

C. Performance Evaluation - Fine Tuned Model

As already mentioned in Section III-C2, we fine-tuned the LLM model used in our RAG system. The fine-tuning was carried out to describe general strategies that we want our model to adopt, as well as to address specific weaknesses that we observed during the development of ICL, giving the model the desired functionality and behavior. At the time of writing, no publicly available version of GPT-4 was available for fine-tuning. To perform fine-tuning, we limited our research to the available model of GPT-3.5. We followed a similar evaluation strategy as in IV-B, where we exposed our system to the same questions (multiple-choice and open-ended).

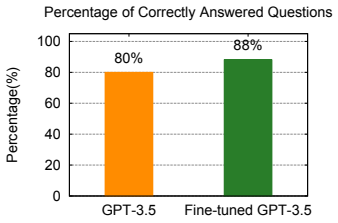


Fig. 5: Raw GPT-3.5 Model vs Fine-tuned GPT-3.5 Model: Percentage of Correctly Answered Questions

Figure 5 shows the percentage of correct answers brought by our RAG system, when utilizing either the standard GPT-3.5 model or the fine-tuned GPT-3.5 Turbo, in combination with the enhanced dataset. As expected, the fine-tuned model achieved 88% correct answers, which suggests a great improvement to the default model, that achieved around 80%. This highlights the necessity of fine-tuning an LLM, in order to fully exploit the model's potential and achieve better performance, especially in the case of specific-domain data. It is worth noting that the fine-tuned GPT-3.5-turbo model does not perform better than the default GPT-4, giving even greater margins of correct answers to a fine-tuned GPT-4 model.

In the case of open-ended responses, we have observed that the fine-tuned GPT-3.5 model was able to answer sophisticated and complex questions with quality reasoning, in contrast to the GPT-3.5 (with enhanced data).

An example of such differences can be seen in Table I. The correct answer should only include nodes equipped with USRP B210s and not those with USRP N210s as illustrated in [19]. The GPT-3.5 provided all the nodes equipped with USRPs, even though not all USRP models are suitable for deploying the 5G Blueprint. Fine-Tuned GPT-3.5 managed to provide only the nodes suitable for conducting the experiment (similar answer to GPT-4 Model).

V. Conclusion & Discussion

In this work, we introduced slAIces, a sophisticated chatbot assistant designed to facilitate and enhance the user experience within SLICES-RI, a complex and large-scale research infrastructure. We presented our methodology and developed a RAG system, employing both In-context learning and Fine-tuning, which represent two popular methods of acquiring special-purpose knowledge to an LLM. Through rigorous testing and evaluation, we concluded that the quality of the chatbot's responses was significantly improved with the data enhancement (GPT-4) and the fine-tuning (GPT-3.5) processes, resulting in accuracy of 96% and 88% in correctly answering multiple-choice questions, respectively.

In the future, we foresee to improve the capabilities of our chatbot by developing an execution multi-agent mechanism, that will automate the experimental workflow and the process of experimentation. This includes several stages such as experiment design, resource reservation, experimentation and result collection. In addition, we aim to develop a closed-loop system that will utilize valid experiment results that have been conducted in the SLICES-RI Infrastructure and subsequently include this data to the main dataset. This will enhance the chatbot's ability to adapt to the needs of the research community, moving beyond tutorial support.

References

[1] S. Fdida *et al.*, "SLICES, a scientific instrument for the networking community," *Computer Communications*, vol. 193, pp. 189–203, 2022.
[2] Y. Demchenko *et al.*, "Slices data management infrastructure for reproducible experimental research on digital technologies," in *2023 IEEE Globecom Workshops (GC Wkshps)*, 2023, pp. 1–6.
[3] Q. Dong *et al.*, "A survey on in-context learning," 2023.
[4] Y. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," 2024.
[5] C. Jeong, "Fine-tuning and utilization methods of domain-specific llms," 2024.
[6] S. Kamnis, "Generative pre-trained transformers (gpt) for surface engineering," *Surface and Coatings Technology*, vol. 466, p. 129680, 2023.
[7] B. Wang *et al.*, "LAVE: LLM-Powered Agent Assistance and Language Augmentation for Video Editing," 2024.
[8] L. Wang *et al.*, "A Survey on Large Language Model based Autonomous Agents," 2024.
[9] X. L. Dong *et al.*, "Towards Next-Generation Intelligent Assistants Leveraging LLM Techniques," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2023.
[10] SLICES-SC, "SLICES Scientific Community Documentation," https://doc.slices-sc.eu/, 2023, accessed: 2024-04.
[11] OpenAI, "Introducing ChatGPT," https://openai.com/blog/chatgpt, accessed: 2024-04.
[12] ——, "GPT-4," https://openai.com/gpt-4, accessed: 2024-04.
[13] J. Liu, "LlamaIndex," 11 2022. [Online]. Available: https://github.com/jerryjliu/llama_index
[14] OpenAI, "Preparing Your Dataset for Fine-Tuning," https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset, accessed: 2024-04.
[15] H. Chen, J. Chen, and J. Ding, "Data Evaluation and Enhancement for Quality Improvement of Machine Learning," *IEEE Transactions on Reliability*, vol. 70, no. 2, pp. 831–847, 2021.
[16] Explosion AI, "en_core_web_lg: English pipeline optimized for CPU." https://spacy.io/models/en#en_core_web_lg, 2023, accessed: 2023-04.
[17] D. Kefalas *et al.*, "slAIces Chatbot Questions Repository," 2024. [Online]. Available: https://repo.nitlab.uth.gr/dkefalas/slaices_questions
[18] N. Makris *et al.*, "Enabling open access to LTE network components; the NITOS testbed paradigm," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, 2015, pp. 1–6.
[19] SLICES Starting Community, "SLICES-RI Blueprint," https://doc.slices-sc.eu/blueprint/, 2023, accessed: 2024-04.