Enhancing Large Language Models for Telecom Networks Using Retrieval-Augmented Generation

Nasik Sami Khan, Md Mahibul Hasan, Md. Shamim Towhid, Saroj Basnet, Nashid Shahriar
Department of Computer Science, University of Regina
{nku618, mhr993, mty754, skb976, nashid.shahriar}@uregina.ca

***Abstract*—This paper presents a comprehensive approach for fine-tuning large language models (LLMs) for domain-specific tasks in the telecommunications field. We utilize a dataset with 1,827 multiple-choice questions (MCQs) from 3GPP standard documents. A publicly available LLM named "Phi-2" is used to answer the MCQs correctly. We develop a Retrieval-Augmented Generation (RAG) pipeline to improve Phi-2 model's performance. The RAG pipeline comprises document segmentation, synthetic question-answer (QA) generation, custom fine-tuning of the embedding model, and incremental fine-tuning of Phi-2. Our experiments show that accuracy greatly increased by combining all the above-mentioned steps in the RAG pipeline. The proposed approach outperforms the baseline by 45.20% in terms of accuracy. This study identifies the limitations of instruction fine-tuning in specialized fields and explores the possibility of using sophisticated data processing with fine-tuned models to improve performance even more.**

***Index Terms*—retrieval-augmented generation, fine-tuning, embeddings, large language models, Telecom, LoRA**

I. Introduction

Large language models' (LLMs) rapid evolution has revolutionized natural language processing (NLP) in numerous domains. However, the use of LLMs in the telecommunications sector has not been extensively implemented, especially in tasks that require specific domain knowledge, such as providing answers to technical questions based on 3GPP standards. Using the TeleQnA [12] dataset, the ITU AI/ML in 5G Challenge brings an opportunity to address this gap by emphasizing on optimizing LLMs for telecom-specific tasks. In this challenge, the task is to utilize either "Phi-2" [1] or "Falcon" [2] to answer the MCQs in the TeleQnA dataset. We design an RAG pipeline that utilizes the "Phi-2" model to generate the answers to the MCQs. The reason behind selecting "Phi-2" is that, it is less resource intensive compared to Falcon. Falcon has seven billion parameters whereas Phi-2 has two billion. The training and test sets are provided on TeleQnA dataset. One restriction on using "Phi-2" is that we cannot fine-tune the model using the options of the MCQs in the training set. A set of 3GPP specifications is shared with us that can be utilized as necessary. These documents contain information that is necessary to answer the MCQs correctly.

The TeleQnA dataset is created by collecting documents from 3GPP standards, research publications, and overview [12]. OpenAI's GPT-3.5 API is utilized to generate synthetic questions from the collected and processed documents. The generated questions go through a human validation process to refine them. Therefore, the generated questions are valid and, at the same time, challenging to answer. To answer the questions, any model must have the domain knowledge. The presence of domain-specific acronyms in the questions and questions with "All of the above" or "None of the above" as options makes the task more practical and challenging.

The study leverages an RAG pipeline to enhance the Phi-2 model's accuracy in answering MCQs. The RAG pipeline is an approach to combining the strengths of the retrieval-based model and the generation-based model to enhance the overall performance of any NLP task [3]. The retrieval model provides context for the generative model. By utilizing the context, the generative model generates the correct output. This RAG approach also helps the generative model to address the well-known hallucination problem [4]. Because of all these advantages of the RAG approach, we design an RAG pipeline to solve this challenge. Any RAG pipeline can be divided into three components: retrieval, augmentation, and generation. We contribute to each of these components in our proposed RAG pipeline. Our main contributions are discussed below.

- We generate QA pairs using the segmented chunks from 3GPP documents and fine-tune the pre-trained embedding model on the generated QA pairs to improve the retrieval process. With this fine-tuning, the embedding model can retrieve related context by which the MCQ can be answered.
- A prompt is carefully designed considering how the "Phi-2" model was originally trained. We augment the prompt with the retrieved chunked documents during the inference.
- To improve the generation process, we fine-tune the "Phi-2" model incrementally on the shared 3GPP documents. This fine-tuned model performs better than the originally trained "Phi-2" which indicates the effectiveness of our incremental fine-tuning process.

The rest of the paper is organized as follows. Our literature survey is discussed in Section II. Section III provides a detailed description of our methodology. All the components of our proposed RAG pipeline are discussed in this section. The results of our proposed approach compared with the selected baseline are presented in Section IV. Continuing our work, the conclusion with our key findings and some future research directions are discussed in Section V.

II. Related Works

Document loading and segmentation are two crucial processes for NLP tasks. Lai et al. introduced a system named LISA which can handle complex, implicit queries by segmentation documents based on user instructions. One of the main capabilities of the tool is that it can produce segmentation from embedding directly. This system demonstrates its zero-shot abilities and robust performances even with limited data for fine-tuning. [5]

Karapantelakis et al. explored the use of LLM for understanding telecommunication standards. They fine-tuned LLMs to handle large and complex documents by providing faster access to relevant information. They also demonstrate how pre-processing as well as segmentation can contribute to increasing the accuracy of a fine-tuned model. [6]

To improve performance of question-answer (QA) models, Alberti et al. developed a technique to generate synthetic QA pairs. The overall process involves generating questions based on segmented text and validating through answer consistency checks. The authors demonstrate how utilizing these synthetic datasets significantly improves the performance of QA models on benchmarks like SQuAD2 and Natural Questions (NQ). [7]

Harris et al. also followed a similar approach of generating synthetic QA pairs to improve the performance of the embedding model. To address the limitation of vocabulary and lack of context, authors use LLMs to rewrite input texts which showed significant improvement in embedding performances on various datasets for embedding model's fine tune. [8]

Zou et al. proposed TelecomGPT, a telecom-specific LLM framework [10]. Authors gathered and prepared pre-training, instruction, and alignment datasets as well as created Telecom Math Modelling, Telecom Open QnA, and Telecom Code benchmarks for evaluation. TelecomGPT surpassed GPT-4, Llama-3, and Mistral in these benchmarks for 3GPP document categorization, telecom code generation, and math modelling in telecommunications.

Zhou et al. surveyed LLMs in telecom and highlighted parameter-efficient fine-tuning (PEFT) methods including low-rank adaptation for fine-tuning big models [9]. The models can be deployed to resource constraint telecom systems to improve efficiency and accuracy of configuration and troubleshooting. Along with PEFT, we needed to follow an incremental learning approach to address resource limits in our training environment.

Our RAG pipeline shares similarities with Josi et al.'s one [14], particularly addressing multimodal data. Unlike their method of converting text, tables, and images into images, we chose to skip the images in both the embedding and fine-tuning. We included the tables only at fine-tuning phase. Our technique ensures predominant behavior of both textual and tabular data and avoids complexity of image processing.

III. Methodology

In this section, we discuss our proposed approaches for answering telecom-specific questions using the RAG pipeline in detail. We divided the main task into six sub-tasks for better understanding. The phases are as follows: (1) Documents Load and Segmentation, (2) Synthetic QA pair Generation, (3) Custom embedding model fine-tuning, (4) Fine-Tuning of the Phi-2 Model, (5) Implementation of the RAG Pipeline, and (6) Answer extraction & post-processing step for result evaluation.

A. Documents Loading and Segmenting

In the first step of the RAG pipeline, we load and segment the raw documents from the 3GPP Release 18 dataset. It contains technical standards related to the telecommunications domain, and the 554 documents were provided in .docx format. We segregate them into more manageable chunks to properly fit into the vector database.

We used the open-source Unstructured library to extract various text elements, such as narrative text, paragraphs, & list items, from the source files. This library helped us parse the documents and relevant metadata, such as the 3GPP release number, which was extracted using regular expressions. Then the documents were loaded and the text divided into smaller, manageable chunks. Each chunk was 100 words in length, a size chosen to ensure that the text segments were compact enough for efficient processing in subsequent stages of our pipeline. For the document chunking, we appended the text to an existing segment or started a new one, depending on the length of the current segment. We also experimented with a 500-token chunk size with the assumption that more context would result in better accuracy in extracting answers for MCQ questions. However, our experiments revealed that the token limit of the Phi-2 model is 2048 tokens. If we provide a larger chunk size for better context, the model fails to generate correct answers during the testing phase. This step for loading documents and separating them into groups made sure that the raw data was handled efficiently and prepared for the next steps in our pipeline. In our data chunking, we skipped the tables and images from the documents.

B. Synthetic QA Generation

We generate synthetic QA pairs with the segmented data from the previous step of our pipeline. These pairs are crucial for fine-tuning the embedding model and for enhancing its ability to accurately process the telecom-specific questions. Each segment from the previous chunks is provided as the context for generating relevant questions. To generate the QA pairs, we designed a prompt template to ensure that each document chunk is provided as an input and the LLM generates a synthetic question from that document chunk. We used the pre-trained Phi-2 model from the Hugging Face pipeline and LangChain framework for this task. We generated a total of 10,000 synthetic QA pairs from the segmented data, instead of creating QA pairs for the whole dataset. Our intuition is that, in the next step of our pipeline, the embedding model will be well-trained with the vocabulary that exists in these 10,000 data rows as they cover a large number of telecom-specific vocabulary. Also, the synthetic QA generation process is computationally expensive and time-consuming to generate for the whole dataset. The generated QA pairs were stored in a CSV file with each row containing an original text segment and its corresponding generated questions.

C. Embedding Model Fine-Tuning

In this step, we focus on fine-tuning a pre-trained embedding model with synthetically generated QA pairs produced in the earlier phase. The main goal is to maximize the performance of the embedding model, especially by adapting vocabularies related to the telecommunication domain so that it manages the domain-specific complexity and nuances robustly.

We divided the 10000 synthetically generated QA data with a 90:10 ratio into training and testing sets to evaluate the model's performance both during and after the fine-tuning process. We used Hugging Face datasets and sentence-transformer libraries for this task. Before the fine-tuning process, we created a baseline result using a pre-trained model, BAAI/bge-base-en-v1.5. This baseline served as a reference point to measure the effectiveness of our fine-tuning results. We evaluated the model using the Normalized Discounted Cumulative Gain (NDCG) metric, which is useful in assessing the quality of retrieval systems. The baseline model was evaluated across multiple embedding dimensions (768, 512, 256, 128, and 64) to provide a comprehensive understanding of its performance at different levels of embedding truncation. This step was vital in assessing the model's ability to execute dimensionality reduction without a substantial decrease in performance.

We used the Matryoshka Representation Learning (MRL) technique [11] to optimize embeddings across various dimensions. The technique is named after the famous Russian game "Matryoshka dolls" in which small dolls are nested within bigger ones. The concept brings a change in the understanding of data representation in the field of AI. This method allows the model to reduce the size of embeddings while retaining crucial information, thus ensuring both accuracy and efficiency.

We implemented a custom loss function, called MatryoshkaLoss, that aggregates loss values across different embedding dimensions. It ensures that the model learns to frontload essential information into the earlier dimensions of the embedding vector. The model produces embeddings at multiple dimensions, and a loss function is applied to both the full-size embeddings and the truncated ones. The loss values from each dimension are combined to create a final loss, which the model minimizes. The model was fine-tuned for 25 epochs on the base model BAAI/bge-base-en-v1, and evaluated on the baseline score to quantify the improvements using the same NDCG score metrics. The fine-tuned model significantly improved retrieval, especially at dealing with complex, domain-specific questions. It demonstrated the advantages of Matryoshka embeddings in balancing performance with storage efficiency. By utilizing truncated embeddings during the initial retrieval phase, the system can quickly narrow down relevant documents or contexts from a large corpus.

D. Fine-Tuning of the Phi-2 Model

In this phase, the focus was on fine-tuning the pre-trained Phi-2 model to enhance its performance, specifically for answering telecom-related questions. The unsupervised fine-tuning process involved several sub-steps, including data preparation, tokenization, model initialization, and the application of advanced fine-tuning techniques to achieve optimal results. We prepared the dataset, ensuring compatibility with the model's architecture. The text data from 554 source documents was first cleaned by removing HTML tags, extra spaces, and other irrelevant characters. Tokenization is performed using a sliding window technique, which is efficient when dealing with larger documents. This approach maintained the inclusion of all important sections of the text during the training process, even if they surpassed the maximum token length. The tokenizer was precisely configured to accommodate the specifications of the Phi-2 model, establishing suitable token lengths and strides to enhance the process. We employed a parameter-efficient fine-tuning method, particularly Low-Rank Adaptation (LoRA). The model was initialized with quantization, which reduces the precision of model parameters, allowing the model to operate more efficiently without sacrificing performance. LoRA is a technique that allows for fine-tuning with a smaller set of parameters, resulting in a substantial reduction in computing expenses while maintaining or improving the model's performance. This technique modifies only a subset of the model's parameters, allowing the model to adapt to the specific requirements of the telecom domain without the need for extensive retraining of the entire model. We used gradient checkpointing and warmup ratios, which are techniques that help stabilize the training process.

Given the computational limitations of our initial servers equipped with NVIDIA RTX A5000 and NVIDIA RTX 3090 GPUs, both having 24 GB of GPU memory, we faced significant delays during the fine-tuning process on the full dataset. Due to the significant duration of the training, we decided to use alternate methods to enhance the efficiency of the procedure. First, we tried with the paid Google Colab Pro platform for the computation, but the session was timed out multiple times. Then finally, we ran our experiments on the Compute Canada server, which is equipped with an NVIDIA A100 GPU featuring 40 GB of GPU memory. Despite the enhanced resources, the amount of the dataset and the complexity of the model still required a more efficient strategy in terms of resource usage. As a result, we adopted an incremental fine-tuning strategy.

This approach involved splitting the training dataset into three subsets and incrementally fine-tuning the model on each subset. Initially, the base Phi-2 model was fine-tuned on the first third of the dataset. This updated model was then used as the starting point for fine-tuning the next third of the dataset. Finally, the process was repeated for the last subset. This stepwise fine-tuning allowed us to manage the large corpus and computational demands effectively. Each phase of fine-tuning on 33% of the dataset took approximately

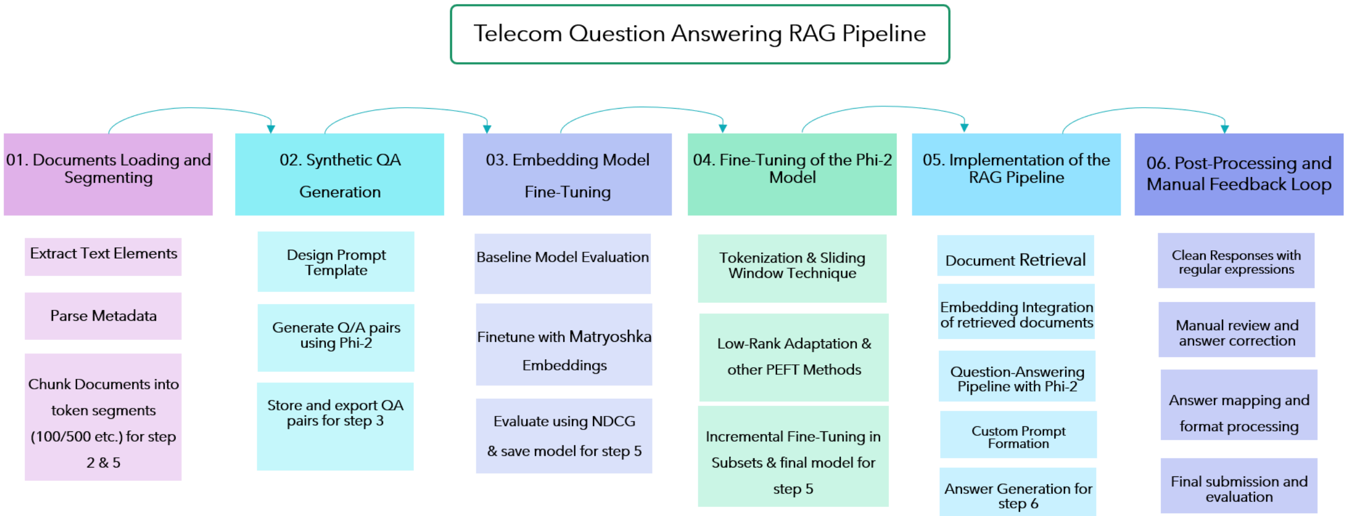

Fig. 1. An overview of the proposed RAG pipeline

one day to complete. This incremental fine-tuning approach provided a practical solution to the computational challenges and contributed to the overall efficiency of the fine-tuning process. We ran our model for 3 epochs, but our experiment showed that only 1 epoch of training was sufficient to get the best result in the competition's evaluation phase, which we will discuss in the result and evaluation section. We also implemented instruction fine-tuning on the dataset, but it did not generate correct answers in most cases, hence resulting in poor performance. Instruction fine-tuning is highly sensitive to the quality and quantity of the instruction and data provided. The use of options of the MCQs for finetuning was restricted. This resulted in a mismatch between the instructions and the actual output of the model and it is one major reason why the model could not generate the output properly.

E. Implementation of the RAG Pipeline

In this step, the fine-tuned Phi-2 model is used to generate answers for multiple-choice questions within a RAG pipeline. The inference process is designed to leverage the strengths of the custom fine-tuned embeddings and the unsupervised fine-tuned Phi-2 model, ensuring accurate and contextually relevant responses. The initial step in the pipeline involved document retrieval and embedding integration. The segmented documents from step 1 in the pipeline were embedded using the fine-tuned model, and these embeddings were stored in a vector database. We used the ChromaDB vector store, which is integrated with the LangChain library, to handle and retrieve these embeddings. This ensured that the retrieval process was highly efficient and capable of rapidly identifying relevant parts of documents in response to a specific query.

The core of the inference process is the question-answering pipeline. We processed the input test data, which was provided in a JSON structure. It contained question ID, question, options, and category value in an MCQ-like pattern. The pipeline is configured to retrieve the most relevant document segments based on the input question. These retrieved documents along with the questions were then passed to the fine-tuned Phi-2 model to generate an answer. A custom prompt template instructed the model to select the correct answer from the provided multiple-choice options. The prompt is stated below:

> *Instruction: You are an AI assistant for answering multiple choice questions from the provided context. You are given the following extracted parts of a long document and a question with some options numbered with capital English letters. Just select the capital English letter of the option that answers the question correctly. No need to explain further.*

This pipeline was effective in handling complex telecom-related queries, as it combined the robust retrieval capabilities of the vector store with the generative abilities of the Phi-2 model. The generated answers are then processed in the next step of the pipeline.

F. Post-Processing and Manual Feedback Loop

The final phase of the pipeline involved post-processing the previous phase's generated answers to improve their correctness and ensure they adhered to the specific format for result submission. This step is crucial for selecting the model's outputs, optimizing overall performance, and preparing the final dataset for submission. Initially, the fine-tuned Phi-2 model's responses were retrieved and cleaned using regular expressions to rigorously refine the answers, while ensuring that only essential information, especially the single letter corresponding to the multiple-choice alternatives (A/B/C/D/E), was preserved. The processes included systematically removing unnecessary content, which resulted in a more streamlined and unified data format. Despite the automated cleaning process, just a small fraction of answers (0.65% to 0.85%) had issues that required manual intervention. For example, the model gave the right responses, but the option number was not indicated in the generated text. Only one to five questions were left unanswered by the model. To deal with these outlier cases, the pipeline includes a manual feedback loop. It included evaluating the results, identifying any remaining errors, and manually fixing them to ensure that each answer followed the expected structure. This iterative method was critical for maintaining high accuracy in the final dataset, especially in situations when the model's output differed from the correct answer. After the answers had been cleaned and verified, they were assigned numeric values (1-5), which were required for the competition's submission format. The use of advanced document retrieval, seamless embedding integration, and rigorous post-processing resulted in the creation of a highly efficient RAG system for retrieving crucial information from large documents.

IV. Results and Evaluation

In this section, we present the findings of our experiments conducted as part of the ITU AI/ML in the 5G Challenge [13]. Our primary focus is to fine-tune the Phi-2 model, fine-tune the embedding model, and implement a RAG pipeline to enhance the model's performance in answering telecom-specific MCQs from the TeleQnA dataset [12]. The dataset contains 1,827 MCQs, and is split into a training set and test set with 1,461 and 366 questions, respectively. The competition also provided 554 supporting documents on 3GPP, and the technical standards related to the telecommunications domain. We performed a series of experiments that involved various strategies for LLM & embedding model fine-tuning, and chunk size optimization to achieve the best accuracy score for the competition. Each submission was evaluated on both the public and private leaderboards, where the public leaderboard measured the performance of 50% of the test set, and the private leaderboard represented the full test set. In the following sections, we discuss the experiment settings and their results.

Evaluation Setting: Table I shows the different configuration settings we considered for our experiments with varying chunk sizes, fine-tuning techniques, and embedding methods. In the first experiment, we considered the pre-trained phi-2 model for generating the answers as a baseline. For the second setting, we explored the instruction finetuned phi-2 model with a finetuned BAAI/bge-small-en-v1.5 embedding model. As the performance improvement was not significant, we tried the custom embedding model with a pre-trained phi-2 model. For all the other compared approaches (4–9), we used the custom embedding model with an unsupervised and incremental finetuned phi-2 model with different document chunk sizes and training epochs.

We used two different chunk sizes, respectively 100 and 500 tokens, to provide a balanced context retrieval while considering the token constraints of the Phi-2 model. The 100-token size provided a suitable amount of context without exhausting the model limit, whereas with the 500-token level in many cases, the model could not generate any answers because of the limitations exhaustion. For the model finetuning,

TABLE I
Compared approaches

Approach	Finetuned Embedding	LLM Model (Phi-2)	Epoch	Chunk Size	Manual Feedback Loop
1. Baseline	×	PT	NA	N/A	×
2. Ins. FT	✓	Ins. FT	5	100	×
3. FT Embedding with PT Phi-2	✓	PT	NA	100	×
4. Inc. FT	✓	Inc. FT	1	100	×
5. Inc. FT	✓	Inc. FT	1	500	×
6. Inc. FT	✓	Inc. FT	2	100	×
7. Inc. FT	✓	Inc. FT	2	500	×
8. Inc. FT with HS	✓	Inc. FT	2	100	×
9. Inc. FT	✓	**Inc. FT**	**1**	**100**	✓

Ins. = Instruction, Inc.= Incremental, PT = Pretrained, FT = Finetuning, HS = Hybrid Search

we implemented an incremental approach and experimented with the model performance with 1 and 2 epochs. Finetuning with 1 epoch was sufficient to provide good results in our experiments. In approach (8), we applied a hybrid search method that combines both vector and keyword-based search mechanisms for context retrieval. The difference between approaches (4) and (9) is that, in the first experiment, the answers generated by LLM were directly used to get the accuracy score. Whereas, in the last experiment, we applied a manual feedback loop to rectify the few incorrect labels generated by LLM. It significantly improved the overall accuracy of the model in our experiments.

Evaluation Results and Discussion: Table II summarizes the results of our key experiments, highlighting the combination of techniques used, and their corresponding performance on the public and private leaderboards.

TABLE II
Evaluation Accuracy of all the approaches

Approach	Public Leaderboard Accuracy	Private Leaderboard Accuracy
1. Baseline	0.2158	0.218
2. Ins. FT	0.3743	0.409
3. FT Embedding with PT Phi-2	0.4645	0.524
4. Inc. FT	0.5519	0.603
5. Inc. FT	0.5355	0.561
6. Inc. FT	0.3798	0.384
7. Inc. FT	0.5301	0.586
8. Inc. FT with HS	0.5846	0.6595
9. Inc. FT	**0.6092**	**0.670**

From Table II, it can be seen that our best-performing approach involved incremental fine-tuning of the Phi-2 model with a 100-token chunk size, which achieved a 67% private leaderboard accuracy, substantially improving the baseline accuracy of 21.8%. This configuration allowed the model to better adapt to the dataset's pattern. The 100-token chunk size was ideal for keeping crucial context without exceeding the model's token processing capabilities, resulting in better retrieval and generation accuracy. The use of MRL was pivotal in improving model performance. By distributing embedding information across multiple dimensions, this approach enabled the pre-trained BAAI/bge-small-en-v1.5 model to efficiently retrieve relevant context and learn the domain-specific vocabulary. The instruction fine-tuning did not perform well in our experiments. The model struggled with telecom-specific instructions, leading to poor results. This outcome demonstrates a limitation in the application of instruction-based fine-tuning within highly specialized domains. In all our experiments, given the input question we retrieved the top 1 matched document as the context from the vector database. Increasing the number of documents retrieved led to the exhaustion of Phi-2's token limit, hence resulting in generating no outputs in most cases.

We also implemented a hybrid search technique that combines vector-based and BM25 retrieval approaches to enhance information retrieval through semantic and lexical matching. This improves coverage, decreases the risk of retrieving semantically related but syntactically irrelevant texts, and provides precise word matching. It is especially useful in specialized sectors where contextual similarity and relevant terminology are both critical. The hybrid method addresses the constraints of vector-based search alone, resulting in a more extensive and accurate retrieval procedure. However, in our experiments, the inference time was twice as long as that of the vector search. This is because two different methods were used simultaneously, resulting in a time-inefficient pipeline given the deadline constraint of the competition.

The baseline results using the pre-trained Phi-2 with the pre-trained BAAI/bge-small-en-v1.5 model served as a benchmark for our experiments. The significant difference between our best result and baseline demonstrates the efficiency of our pipeline in greatly enhancing the performance of the model.

V. Conclusion & Future Works

The goal of this study is to improve the Phi-2 model's performance in the field of telecommunications. Our best-performing model configuration reached a 67% accuracy on the private leaderboard, improving the baseline score by 45.20%. Significant improvements in accuracy are achieved by fine-tuning the pre-trained Phi-2 model and using MRL for embedding finetuning. The incremental fine-tuning technique proved efficient in managing the computational constraints, which resulted in a feasible solution for this task.

Future work could focus on including diverse document formats like summaries of tables, and image descriptions through a multi-modal RAG pipeline, which could enhance the model's performance. Furthermore, instruction fine-tuning for telecom-specific tasks, exploring other larger embedding models, handling complex queries with sophisticated RAG pipeline frameworks, and use of advanced prompt engineering techniques could be explored. This study improves the performance of LLM and implements the RAG pipeline for telecom domain solutions. Our proposed methods have the potential to be applied in other specialized areas such as cybersecurity, healthcare, law, or finance, where they can enhance general-purpose LLMs by fine-tuning them to meet the unique demands of each field.

References

[1] Javaheripi, M. and Bubeck, S. (2023) "Phi-2: The surprising power of small language models, Microsoft Research." (Accessed: 20 August 2024).

[2] E. Almazrouei et al., "Falcon-40B: an open large language model with state-of-the-art performance", 2023.

[3] Gao, Yunfan, et al. "Retrieval-augmented generation for large language models: A survey." arXiv preprint arXiv:2312.10997 (2023).

[4] Li, Jiarui, Ye Yuan, and Zehua Zhang. "Enhancing llm factual accuracy with rag to counter hallucinations: A case study on domain-specific queries in private knowledge-bases." arXiv preprint arXiv:2403.10446 (2024).

[5] X. Lai et al., "LISA: Reasoning Segmentation via Large Language Model," May 01, 2024, arXiv: arXiv:2308.00692. doi: 10.48550/arXiv.2308.00692.

[6] A. Karapantelakis et al., "Using Large Language Models to Understand Telecom Standards," Apr. 12, 2024, arXiv: arXiv:2404.02929. doi: 10.48550/arXiv.2404.02929.

[7] C. Alberti, D. Andor, E. Pitler, J. Devlin, and M. Collins, "Synthetic QA Corpora Generation with Roundtrip Consistency," Jun. 12, 2019, arXiv: arXiv:1906.05416. doi: 10.48550/arXiv.1906.05416.

[8] N. Harris, A. Butani, and S. Hashmy, "Enhancing Embedding Performance through Large Language Model-based Text Enrichment and Rewriting," Apr. 18, 2024, arXiv: arXiv:2404.12283. doi: 10.48550/arXiv.2404.12283.

[9] H. Zhou et al., "Large Language Model (LLM) for Telecommunications: A Comprehensive Survey on Principles, Key Techniques, and Opportunities," May 17, 2024, arXiv: arXiv:2405.10825. doi: 10.48550/arXiv.2405.10825.

[10] H. Zou et al., "TelecomGPT: A Framework to Build Telecom-Specfic Large Language Models," Jul. 12, 2024, arXiv: arXiv:2407.09424. doi: 10.48550/arXiv.2407.09424.

[11] "Introduction to Matryoshka Embedding Models." Accessed: Aug. 12, 2024. [Online]. Available: https://huggingface.co/blog/matryoshka

[12] A. Maatouk, F. Ayed, N. Piovesan, A. De Domenico, M. Debbah, and Z.-Q. Luo, "TeleQnA: A Benchmark Dataset to Assess Large Language Models Telecommunications Knowledge," Oct. 23, 2023, arXiv: arXiv:2310.15051. Accessed: Aug. 16, 2024. [Online]. Available: http://arxiv.org/abs/2310.15051

[13] Zindi, "Specializing Large Language Models for Telecom Networks," Zindi. Accessed: Aug. 17, 2024. [Online]. Available: https://zindi.africa/competitions/specializing-large-language-models-for-telecom-networks

[14] P. Joshi, A. Gupta, P. Kumar, and M. Sisodia, "Robust Multi Model RAG Pipeline For Documents Containing Text, Table & Images," in 2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), Jun. 2024, pp. 993–999. doi: 10.1109/ICAAIC60222.2024.10574972