# Efficiency of LLMs for Understanding 3GPP Specifications Using Approaches such as RAG and Fine Tuning, and Evaluation with Accuracy and RAGAS

José de Arimatéa Passos Lopes Júnior

17 November 2024

**Abstract**

This work aims to evaluate the performance of small and large language models (LLMs) in answering questions related to telecommunications, leveraging two main datasets: TeleQnA, a question-and-answer set specific to telecommunications, and TSpec-LLM, an extensive repository of technical documents from multiple 3GPP releases. A Retrieval-Augmented Generation (RAG) system was implemented to enhance response accuracy by incorporating relevant context from TSpec-LLM into the TeleQnA queries. This setup enables a comparative analysis of the performance of small and large language models, with a focus on assessing whether fine-tuning a smaller model on the telecommunications-specific dataset improves its performance.

## 1 Introduction

The growing use of Large Language Models (LLMs) has been transforming various fields of knowledge, including telecommunications, where technical complexity and the constant evolution of standards and specifications demand models capable of understanding and interpreting extensive and detailed technical documents. In this context, the evaluation and improvement of models that can assist in interpreting technical standards and answering specific questions in telecommunications become relevant, especially to ensure the correct and precise understanding of information contained in 3GPP (3rd Generation Partnership Project) normative documents.

The objective of this work is to compare the performance of different large language models (LLMs) in answering questions from a specific dataset, TeleQnA, a set of questions and answers specifically designed for telecommunications. Additionally, the TSpec-LLM dataset, an extensive repository of 3GPP technical documents covering multiple Releases, was utilized as a basis for implementing a Retrieval-Augmented Generation (RAG) system to enrich the context of responses to TeleQnA questions, enabling a comparative analysis of the performance of three distinct language models. Furthermore, the impact of fine-tuning a smaller model will also be evaluated to verify whether this adaptation to the specific telecommunications domain significantly improves its performance, as was done in [7].

The experiments were conducted to evaluate the ability of the models in selecting correct answers and generating content based on questions from 3GPP Release 17, testing both the models' ability

to answer direct questions and their precision when using RAG to retrieve and process additional information from TSpec-LLM. The evaluation metrics applied include accuracy, when answer options are provided, and a set of specific metrics from the Retrieval-Augmented Generation Assessment Suite (RAGAS) for generated responses without options, aiming to provide a comparative analysis between the models.

# 2 Dataset

Two different datasets were used in this work: TeleQnA and TSPEC-LLM.

## 2.1 TeleQnA

TeleQnA[3] is a telecommunications-focused dataset containing 10,000 questions and answers extracted from various sources, including standards and research articles. Organized into five main categories — *research publications*, *research overview*, *standards specifications*, *standards overview*, and *lexicon* — the dataset offers comprehensive coverage of telecommunications knowledge. With approximately 6 million words across 25,000 pages, TeleQnA provides a detailed representation of both technical and general topics in the field, serving as an essential tool for assessing understanding and proficiency in standards, research, and domain-specific terminology.

Each item in the dataset[10] consists of a question, options, the correct answer, and an explanation of the answer. An example is shown below:

**Question 1**: What advantages does a data center-enabled HAP (High Altitude Platform) offer from an energy perspective?

- Option 1: Saves cooling energy and harvests solar energy
- Option 2: Uses renewable energy and offsets carbon emissions
- Option 3: Deploys large-scale solar panels and batteries
- Option 4: Requires less energy for communication links

**Answer**: Option 1: Saves cooling energy and harvests solar energy
**Explanation**: The data center-enabled HAP saves cooling energy due to its location at the naturally low-temperature stratosphere and harvests solar energy using large solar panels.
**Category**: Research publications

TeleQnA was used to train the *Unsloth* model[13] and to conduct the comparative testing among the three models presented in this work.

For testing, only questions related to Release 17 of 3GPP were used, as it is the most recent and complete release in the TSpec-LLM dataset[8]. This approach allows for a more accurate evaluation of RAG's efficiency since it is more likely that the context of a TeleQnA question exists in the TSpec-LLM dataset and can therefore be retrieved using RAG. The number of questions related to Release 17 in the TeleQnA dataset is 733, spanning only the *standards specifications* and *standards*

*overview* categories.

The test set consisted of 100 questions, with 50 from the *standards specifications* category and 50 from the *standards overview* category.

### 2.2 TSpec-LLM

TSpec-LLM[9] is a comprehensive and open-source dataset developed to assist language models in understanding technical telecommunications documents, especially those produced by the 3rd Generation Partnership Project (3GPP). This dataset covers all 3GPP specification documents, from Release 8 to Release 19 (1999–2023), preserving the original content, including tables and formulas essential for the technical context. Comprising more than 30,000 documents and approximately 535 million words, TSpec-LLM enables Large Language Models (LLMs) to effectively analyze and organize complex telecommunications information, facilitating the interpretation of standards specifications. TSpec-LLM is therefore an essential tool for research in AI applied to telecommunications, allowing LLMs to be used as assistants in reading and analyzing industry technical standards.

TSpec-LLM extracts the text from 3GPP documents and formats it in Markdown[8], simplifying the manipulation of information and facilitating data extraction for use in Machine Learning models.

The TSpec-LLM dataset will be used to assist in information retrieval within the Retrieval-Augmented Generation (RAG) framework. In this context, the dataset will serve as the database of 3GPP technical documents, enabling the retrieval system to locate and provide relevant excerpts from these documents to complement the responses generated by LLMs.

## 3 Methodology

The project was developed in Python and can be accessed on the GitHub repository at [6].

### 3.1 Models

Three different models were tested for comparison using the TeleQnA dataset: the GPT-4o-mini model from OpenAI, the Llama 3.2 model with 3 billion parameters from Meta, utilized through the Unsloth library[13], and the same Llama 3.2 model with 3 billion parameters but fine-tuned[11] and trained using the TeleQnA dataset[10].

### 3.2 Unsloth

The *Unsloth*[13] is an open-source repository that provides various LLM models and features a well-defined structure that facilitates loading the model onto a GPU and subsequently training the model in a quick and practical manner.

### 3.3 Fine Tuning

The fine-tuning of the Llama 3.2 model with 3 billion parameters followed the structure outlined in [12].

### 3.3.1 Training Dataset

The data used for training came from the TeleQnA dataset[10], in which 4,000 questions were utilized. Of these 4,000 questions, 500 were related to Release 17 (leaving 233 for testing), while the remaining 3,500 questions were about other Releases, excluding Releases 17 and 18.

For the training, two different pairs of context and expected response were provided:
In the first 2,000 questions, the pairs were:

- Context: question and options
- Expected Response: the correct option and the text of the correct option

In the last 2,000 questions, the pairs were:

- Context: only the question
- Expected Response: only the text of the correct option

Finally, the questions were shuffled.

### 3.3.2 Parameters and Training

The training code can be found in [6], under the path "*/Source/Fine_tuning/*". Due to personal hardware limitations (the local GPU is an NVIDIA GeForce RTX 3050 Ti with a maximum memory of 3.712 GB), the model training was conducted on Google Colab. This setup aimed to load the model without 4-bit quantization and train it with more questions in less time.

The model quantized to 4 bits occupies 2.768 GB of GPU memory. Without 4-bit quantization, the model loaded from Unsloth occupies 6.688 GB, with a peak memory usage of 8.658 GB during training with the 4,000 questions.

The training parameters were:

- Number of trainable parameters: 24,313,856
- Batch size per device: 2
- Gradient accumulation steps: 16
- Batch size: 32
- Total steps: 400
- Number of epochs: 3.2
- Learning rate: 0.0002
- Optimizer: AdamW
- Initial loss: 3.207
- Final loss: 0.333

The training lasted approximately 42 minutes, resulting in the model referred to as Llama 3.2 3B Lora, due to the addition of Lora layers caused by the Fine Tuning.

### 3.4 RAG

RAG (Retrieval Augmented Generation) is a retrieval system that searches for relevant excerpts in a specific dataset or document collection, providing the language model with additional context to enhance the accuracy and relevance of the generated response.

In this work, the documents used for information retrieval were from the TSpec-LLM Dataset[8].

First, the documents were divided using the *"MarkdownHeaderTextSplitter"*, creating 780,651 chunks, each with a size of 2,000 characters and a 100-character overlap between chunks. The Embeddings model used was "*SentenceTransformer('all-mpnet-base-v2')*".

Subsequently, the Embeddings were indexed using Faiss, generating the file "faiss_index.bin", which was later used to perform searches based on cosine similarity. The input for RAG searches consisted of questions with options for accuracy tests and questions alone for RAGAS metrics evaluation.

## 4 Experiments

The models GPT-4o-mini, Llama 3.2 3B, and Llama 3.2 3B lora were evaluated in two different ways using 100 questions from the TeleQnA Dataset related to Release 17, with the question prompt employing Chain of Thought, instructing the model to think step by step before answering.

In the first evaluation, the question was presented along with multiple-choice options. The models were required to select the correct option, and the evaluation metric used was accuracy.

In the second evaluation, the question was presented without options, and the models had to provide an open-ended response. The evaluation method involved metrics from RAGAS (Retrieval-Augmented Generation Assessment Suite). Some of the RAGAS metrics depended on the assessment of a LLM (GPT-4o-mini was used for this evaluation), while others did not rely on an LLM.

- Metrics with LLM: Factual Correctness, Semantic Similarity, and Rubrics Score with reference;
- Metrics without LLM: Bleu Score, Rouge Score, Exact Match, and String Present.

### 4.1 Accuracy

The prompts used for questions with options were as follows:

- Without RAG:

```
f"Question: {question}\n"
f"Options:\n" + "\n".join(options) + "\n"
"Think step by step before answering and respond with the correct
option in the format 'correct option: <X>'."
```

- With RAG:

```
f"Relevant Information:\n{rag_results}\n"
f"Question: {question}\n"
f"Options:\n" + "\n".join(options) + "\n"
"Think step by step and choose the correct option.\n"
"You must respond in the format 'correct option: <X>', where <X>
is the correct letter for the option."
```

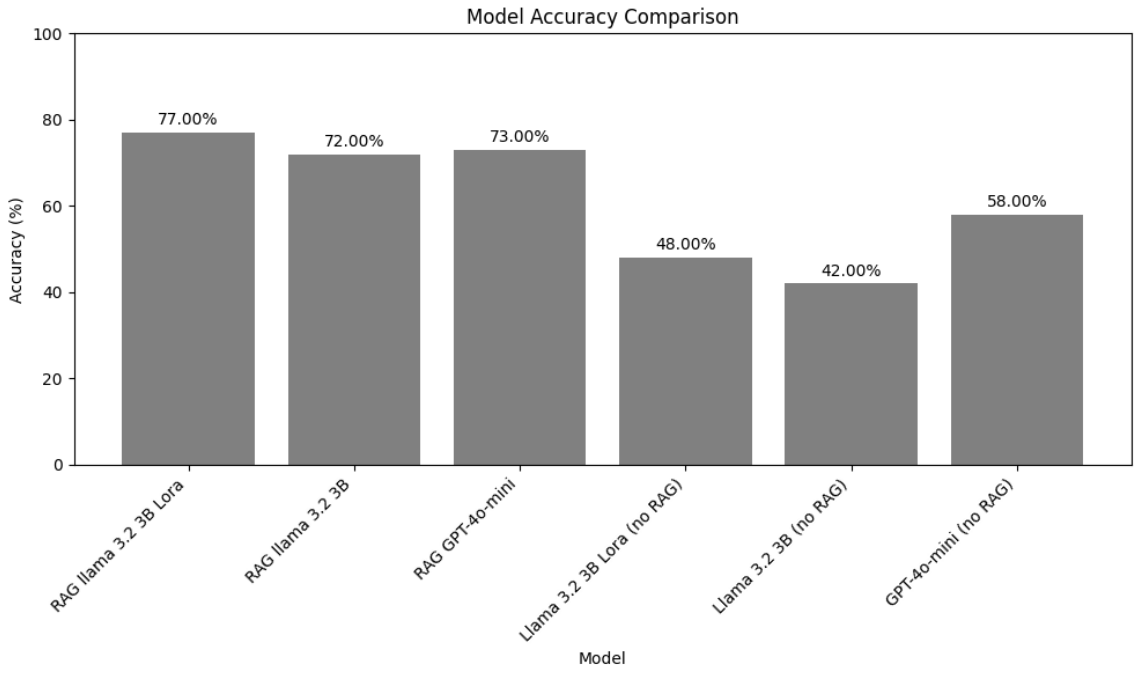


Figure 1: Accuracy evaluation of the models answering 100 multiple-choice questions.

Figure 1 presents the results obtained by the models when answering 100 multiple-choice questions, requiring them to select the correct option. The values expressed as percentages represent the accuracy rate of each model.

## 4.2 RAGAS

The prompts used for the questions without options were:

- Without RAG:

  ```
  f"Question: {question}\n"
  "Think step by step before answering and respond with a final answer in the format 'answer:
  <XXXXX>'."
  ```

- With RAG:

```
f"Relevant Information:\n{rag_results}\n"
f"Question: {question}\n"
"Think step by step before answering and analyze the relevant information carefully, then
respond with a final answer in the format 'answer: <XXXXX>'."
```

| **Models** | **Factual Correctness Values (0 to 5)** | **Semantic Similarity Values (0 to 1)** | **Rubrics Score with reference Values (0 to 1)** |
|---|---|---|---|
| RAG llama 3.2 3B Lora | 0,180 | 0,410 | 1,810 |
| RAG llama 3.2 3B | 0,141 | 0,415 | 2,370 |
| RAG GPT-4o-mini | 0,181 | 0,421 | 2,530 |
| Llama 3.2 3B Lora (no RAG) | 0,008 | 0,349 | 1,530 |
| Llama 3.2 3B (no RAG) | 0,078 | 0,500 | 1,950 |
| GPT-4o-mini (no RAG) | 0,061 | 0,362 | 2,240 |

Table 1: Results of the models in terms of Factual Correctness, Semantic Similarity, and Rubrics Score with reference

Table 1 presents the results obtained by the models answering the questions without options and evaluating them using RAGAS metrics that require a LLM, in this case, GPT-4o-mini.

The *Rubrics Score* with reference is a custom metric where the score is assessed on a scale from 1 to 5. The reference for the metrics is provided to the LLM for evaluation as follows:

```
rubrics = {
"score1_description": "The response is incorrect, irrelevant, or does not align with the
ground truth.",
"score2_description": "The response partially matches the ground truth but includes
significant errors, omissions, or irrelevant information.",
"score3_description": "The response generally aligns with the ground truth but may lack
detail, clarity, or have minor inaccuracies.",
"score4_description": "The response is mostly accurate and aligns well with the ground
truth, with only minor issues or missing details.",
"score5_description": "The response is fully accurate, aligns completely with the ground
truth, and is clear and detailed.",
}
```

| Models | Bleu Score Values (0 to 1) | Rouge Score Values (0 to 1) | Exact Match Values (0 to 1) | String Present Values (0 to 1) |
|---|---|---|---|---|
| RAG llama 3.2 3B Lora | 0,066 | 0,243 | 0,060 | 0,070 |
| RAG llama 3.2 3B | 0,058 | 0,200 | 0,050 | 0,050 |
| RAG GPT-4o-mini | 0,039 | 0,197 | 0,040 | 0,040 |
| Llama 3.2 3B Lora (no RAG) | 0,021 | 0,183 | 0,020 | 0,020 |
| Llama 3.2 3B (no RAG) | 0,056 | 0,174 | 0,000 | 0,000 |
| GPT-4o-mini (no RAG) | 0,003 | 0,139 | 0,000 | 0,020 |

Table 2: Results of the models in terms of Bleu Score, Rouge Score, Exact Match, and String Present

Table 2 presents the results obtained by the models answering the questions without options, evaluated using RAGAS metrics that do not require a LLM for assessment.

# 5 Conclusion

## 5.1 Accuracy

The model's training demonstrated significant improvements in performance, both with and without RAG (Retriever-Augmented Generation). With the use of RAG, the model was able to better understand the context and provide more accurate answers, even surpassing GPT-4o-mini and achieving impressive results. The tests conducted with RAG proved promising, exceeding the results presented in the paper on TeleQnA[3], even though the dataset used was smaller compared to the original study.

Despite limited training, progress in the model's performance was observable. With a larger dataset, it is possible that the model could improve even further. Additionally, the use of RAG showed that an excessively complex model is not required to achieve satisfactory results, as this approach alone already delivers very good performance.

## 5.2 RAGAS

The overall scores were low, but the focus was not on evaluating the absolute performance of the models, as the metrics can be subjective, especially those that rely on LLM evaluations. The main goal was to compare the results of the models. In this regard, the models with RAG stood out, presenting superior results.

However, without the options, GPT-4o-mini performed better overall compared to the other models, especially in the LLM evaluation, with answers that were closer to the correct ones. On the other hand, in the evaluations without LLM, Llama 3.2 3B with Fine-Tuning achieved better results, with answers aligning more with the correct way of writing, reflecting the training focused on more precise words and more objective responses.

It is important to note that although Llama 3.2 lora provided more objective answers, this did not necessarily indicate a higher number of correct responses. The other models, despite more varied

or less objective answers, also answered questions correctly in equal measure.

### 5.3 General Conclusions

This work demonstrates that with the use of RAG, small models like Llama 3B can achieve very good performance. Additionally, it is important to highlight that during inference, in the tests with 100 questions, the model was loaded with 4-bit quantization so that it did not occupy a large memory space (Section 3.3.2). Therefore, even with a small and quantized model, the results were similar to those of GPT-4o-mini, both utilizing RAG. The Fine-Tuned model still achieved superior accuracy results compared to GPT-4o-mini (Section 4.1). The results were also better than those presented in the article [3], considering that the 100 questions tested were only from the standards specifications and standards overview categories, whereas the test set in [3] was much larger.

These results are significant in demonstrating that applications requiring technical or specific knowledge do not need a large and expensive model to perform such tasks. It is possible to achieve similar or even superior performance with smaller, more affordable models that can run locally, as they occupy a relatively small amount of memory space, especially since model quantization did not lead to a drop in performance.

It is also worth noting that the time spent on inference for the smaller models was not significantly higher than that of GPT-4o-mini. Although not shown in this work, inference for each question took only a few seconds to provide a response. No round of 100 questions took more than 20 minutes.

## 6 Agents

Tests were conducted using LLM model Agents to evaluate their performance in comparison to the results obtained with RAG. The Agent had a search tool, which was the same RAG used previously.

The models used as Agents were the Llama 3.2 3B from Ollama and the Llama 3.1 70B provided by the company Venturus, based in Campinas-SP, in partnership with the IA024 course at Unicamp, taught by Professor Roberto Lotufo in 2024.

The Langchain ReAct library[2] was attempted to produce the Agent, but the models did not perform well with some prompt attempts, especially the smaller Llama 3.2 3B model.

Thus, a chain of functions was created using the models to generate a custom agent for this specific work, based on the code in [1]. The Agent's code can be found in [6], in the "*Source/Agent/*" path. The Agent's tool was the RAG, which retrieved the 3 most relevant pieces of information.

The results using Agents are shown in Table 3:

| Models | Acccuracy Values (0 a 100) | Number of None Responses Values (0 a 100) |
|---|---|---|
| Agent llama 3.2 3B | 56% | 5 |
| Agent llama 3.1 70B Venturus | 56% | 7 |

Table 3: Agents results in terms of Accuracy and Number of Unanswered Questions (None)

The results are superior to the tests without RAG, but worse than when using only RAG with the retrieval of the 5 most relevant contexts. Since the same RAG was used as the tool, it was expected that the results would be close to those in section 4.1, and even when forcing the Agent to perform at least one search, the models still made more mistakes than expected. It is believed that the prompt might be improved, or the agent's action chain could be optimized to yield better results. The performance drop is not due to a reduction in the number of relevant pieces of information provided by the RAG, as tests were conducted with only 3 contexts instead of 5, and the results were similar to those in section 4.1.

# 7 Future Work

The results presented in this work are very promising for small models, whether trained or not. However, with more comprehensive studies and new solutions emerging every day, it is expected that the performance of future models will be even better.

For future work, it would be interesting to further explore the use of agents for answering questions, aiming to improve the quality of the search and the step-by-step answering process. Additionally, improvements in the use of RAG could be explored, such as the creation of smaller *chunks*, the implementation of more efficient *embeddings*, or the application of a new preprocessing technique to eliminate irrelevant information from documents.

With a more robust hardware infrastructure, it would be feasible to conduct training with a larger *dataset*, as well as explore the comparison between models of different sizes, observing the impacts of training in each configuration.

Another possibility is to test the performance of the search on local documents and the internet, evaluating how each approach impacts the quality of the answers. This approach is similar to the one used in Telco-RAG in [4][5]. Furthermore, a comparison could be made between the results obtained in the models from this work and those in the Telco-RAG models[4].

# References


[1] How to create your own llm agent from scratch: A step-by-step guide. https://medium.com/@incle/how-to-create-your-own-llm-agent-from-scratch-a-step-by-step-guide-14b763e5b3b8. Accessed: Nov 2024.

[2] React. https://python.langchain.com/v0.1/docs/modules/agents/agent$_t ypes/react/.Accessed$ : $Nov$2024.

[3] Nicola Piovesan Antonio De Domenico Merouane Debbah Zhi-Quan Luo Ali Maatouk, Fadhel Ayed. Teleqna: A benchmark dataset to assess large language models telecommunications knowledge. *arXiv preprint arXiv:2310.15051*, 2023. Accessed: Nov 2024.

[4] Antonio De Domenico Nicola Piovesan Ali Maatouk Andrei-Laurentiu Bornea, Fadhel Ayed. Telco-rag: Navigating the challenges of retrieval augmented language models for telecommunications. *arXiv preprint arXiv:2404.15939*, 2024. Accessed: Nov 2024.

[5] Mckay Wrigley Andrei-Laurentiu Bornea. Telco-rag repository. https://github.com/netop-team/Telco-RAG. Accessed: Nov 2024.

[6] José Arimatea. 3gpp llm evaluation. https://github.com/josearimatea/3gpp_llm_evaluation. Accessed: Nov 2024.

[7] Alexandros Nikou Farnaz Moradi Christian Olrog Fitsum Gaim-Henrik Holm Doumitrou Daniil Nimara Vincent Huang Athanasios Karapantelakis, Mukesh Thakur. Using large language models to understand telecom standards. *arXiv preprint arXiv:2404.02929*, 2024. Accessed: Nov 2024.

[8] Rasoul Nikbakht. Tspec-llm repository. https://huggingface.co/datasets/rasoul-nikbakht/TSpec-LLM. Accessed: Nov 2024.

[9] Mohamed Benzaghta Rasoul Nikbakht and Giovanni Geraci. Tspec-llm: An open-source dataset for llm understanding of 3gpp specifications. *arXiv preprint arXiv:2406.01768*, 2024. Accessed: Nov 2024.

[10] Netop Team. Dataset teleqna. https://huggingface.co/datasets/netop/TeleQnA. Accessed: Nov 2024.

[11] UnsLoth. How to finetune llama-3 and export to ollama. https://docs.unsloth.ai/tutorials/how-to-finetune-llama-3-and-export-to-ollama. Accessed: Nov 2024.

[12] Unsloth. Llama 3.2 fine tuning. https://colab.research.google.com/drive/1T5-zKWM_5OD21QHwXHiV9ixTRR7k3iB9?usp=sharing. Accessed: Nov 2024.

[13] Unsloth. Unsloth repository. https://github.com/unslothai/unsloth?tab=readme-ov-file. Accessed: Nov 2024.