HOCHSCHULE OSNABRÜCK
UNIVERSITY OF APPLIED SCIENCES

Fine-Tuning Large Language Models with Low-Rank Adaptation: An Evaluation of Performance and Efficiency

Thomas Adelt–September 10, 2025

Preamble

In the preparation of this work, various artificial intelligence tools, including *Perplexity*, *Consensus*, *SciSpace*, and *ScholarAI*, were employed to support the research process. These tools were utilized to assist in identifying relevant literature, summarizing findings, and exploring multiple perspectives on the topics discussed.

As English is not the first language of the author, AI-based language tools were also used to enhance the fluency, clarity, and readability of the text. This assistance was limited to refining grammar, syntax, and style without altering the underlying meaning or originality of the ideas. An illustrative example of such improvements is provided in the Appendix-Section A.

Additionally, AI tools were used to assist in the development, debugging, and optimization of code relevant to this work.

Abstract

This study investigates the performance and efficiency of Low-Rank Adaptation (LoRA) as a parameter-efficient fine-tuning method for injecting knowledge into large language models (LLMs) in the context of academic and career counseling. A foundation model was fine-tuned with LoRA on an AI-generated domain-specific dataset and compared to both the baseline and a Retrieval-Augmented Generation (RAG) pipeline. Evaluation combined LLM-as-a-judge metrics, general capabilities benchmarks, and efficiency indicators such as training time, inference latency, and VRAM usage. Results show that LoRA-enhanced models generally outperform the baseline in answer relevancy, hallucination reduction, and factual correctness, though performance varies across configurations and sometimes even declines in factual correctness and general capabilities benchmarks. Training is highly efficient, requiring only $\approx$ 30 minutes on a single GPU, but dataset creation introduces significant overhead. RAG achieved stronger factual accuracy, faithfulness, and output stability than LoRA-models, albeit with higher inference latency and memory demands. LoRA proved particularly effective for domain-specific contextual alignment, shown by the answer relevancy metric, whereas RAG excelled in factual grounding and robustness. These findings highlight the trade-offs and complementary advantages of both methods, suggesting that the choice between LoRA and RAG should depend on application-specific priorities concerning accuracy, efficiency, and resource constraints.

Keywords: Large Language Models, Low-Rank Adaptation, Retrieval-Augmented Generation, Fine-Tuning, Parameter-Efficient Fine-Tuning, Evaluation

1 Introduction

Large Language Models (LLMs) have revolutionized the field of natural language processing, enabling machines to understand and generate human-like text in large contexts. However, the computational resources required to train and fine-tune these models are substantial, often limiting their accessibility [36, p. 179]. This study explores the use of Low-Rank Adaptation (LoRA) by Hu et al. [16] as a method to fine-tune LLMs efficiently, focusing on its performance and efficiency in domain-specific knowledge injection.

1.1 Background and Context

As described in the introduction, the field of Natural Language Processing (NLP) has been significantly influenced by LLMs as machines are now able to generate contextually coherent and human-like text [36, p. 10]. Their capabilities, however, come at the cost of substantial computational requirements, which limit their practical deployment, especially in resource-constrained environments. Traditional fine-tuning methods involve updating all parameters of a large model, leading to high memory and compute costs [36, p. 179].

LoRA addresses these limitations by injecting trainable low-rank matrices into existing weights, thus enabling parameter-efficient adaptation without compromising performance [16, p. 1-2]. This approach has made fine-tuning LLMs more feasible for smaller institutions or specialized applications. As a case study, this work applies LoRA to the domain of academic and career counseling. The goal is to investigate whether a LoRA-adapted LLM can provide relevant, accurate, and context-aware responses to support domain-specific tasks.

1.2 Research Question and Objectives

The aim of this study is to fine-tune an LLM using LoRA to enable domain specific knowledge injection in a resource-efficient manner. LoRA reduces the number of trainable parameters by introducing low-rank updates, thereby maintaining performance while lowering computational overhead. The adapted model is evaluated on its ability to function within the domain of career counseling, focusing on accuracy, factuality, and domain relevance of its outputs.

Efficiency is assessed in terms of hardware resource utilization and temporal efficiency—for all aspects of the pipeline—while performance is often evaluated using traditional metrics such as ROUGE [24], but in this study it is focussed, but not limited, on LLM-as-a-judge metrics like hallucination and factual correctness. To benchmark the effectiveness of LoRA, results are compared with a non-fine-tuned (foundation) model and a Retrieval-Augmented Generation (RAG) [21] pipeline.

The guiding research question is: *How can Low-Rank Adaptation be used to fine-tune Large Language Models for domain specific knowledge injection, and how does it compare to other techniques like Retrieval-Augmented Generation in terms of performance and efficiency?*

1.3 Literature Review

The literature review will describe the foundational theories and concepts that underpin the research, including the theoretical foundations of transformer architecture, parameter-efficient fine-tuning methods, and the specific contributions of LoRA to position it in the field.

1.3.1 Theoretical Foundations and Transformer Architecture

The seminal paper by Vaswani et al. [50] introduced the transformer model based solely on attention mechanisms, establishing the architectural foundation upon which LoRA operates [16, pp. 4–5]. This paper demonstrated that attention mechanisms could replace recurrent and convolutional components entirely, achieving superior performance in machine translation tasks while enabling parallel processing [50, pp. 2, 10].

Following the transformer architecture, scaling laws for language models provide crucial theoretical context. The work on these scaling laws by Kaplan et al. [19] demonstrates the relationship between model size, data, and performance, establishing that larger models consistently achieve better performance [19, p. 19]. These scaling laws create the fundamental motivation for parameter-efficient methods like LoRA, as full fine-tuning of increasingly large models becomes computationally prohibitive.

The parameter efficiency of LoRA arises from the assumption that the updates required for adapting a pre-trained model lie in a "low intrinsic rank" subspace of the full parameter space [16, p. 2], [1, p. 8]. In particular, a standard dense weight matrix $W \in \mathbb{R}^{d\times k}$ contains $d \cdot k$ trainable parameters [50, p. 5], whereas LoRA replaces this with two low-rank matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, yielding

$$\#\text{params}_{\text{LoRA}} = r(d + k) \ll d \cdot k \,, \tag{1}$$

where r denotes the LoRA-rank, d for the models dimension, and k for the input dimension. Since $r \ll \min(d, k)$, this results in a substantial reduction in trainable parameters [16, p. 4].

Building on this principle, LoRA constrains the update for a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$ through a low-rank decomposition

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

where the frozen matrix W_0 is complemented by the trainable update $\Delta W = BA$ with A initialized from a Gaussian distribution and B set to zero [16, p. 4]. The forward pass is thereby modified to

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

with $\Delta W x$ scaled by $\frac{\alpha}{r}$ to regulate the adaptation strength [16, p. 4].

1.3.2 Parameter-Efficient Fine-Tuning Methods

To position LoRA in the broader Parameter-Efficient Fine-Tuning (PEFT) landscape, it is essential to present several complementary approaches. Early developments such as adapter-based methods introduced small trainable modules between transformer layers while keeping the original parameters frozen, showing that updating only a fraction of parameters can achieve performance comparable to full fine-tuning and thus laying the foundation for subsequent PEFT methods [13, pp. 6, 19]. Building on this idea, prefix and prompt tuning approaches prepend trainable vectors to the keys and values of attention layers, enabling adaptation to new tasks without altering the underlying weights [23, pp. 3–5], with extensions such as P-Tuning v2 showing that well-optimized prompt tuning can match the performance of full fine-tuning across different model scales [28,

p. 5]. The relationship between prefix tuning and adapter methods has been further clarified by inducer-tuning, which leverages residual structures from adapter tuning to overcome initialization challenges in prefix tuning [7, p. 9]. Within this evolving landscape, LoRA has given rise to a range of important variants and extensions: QLoRA incorporates quantization techniques to enable efficient fine-tuning with 4-bit precision while preserving performance [11, pp. 4–5, 16], AdaLoRA introduces adaptive rank allocation to dynamically adjust ranks according to task importance, thereby addressing the limitations of fixed-rank LoRA [56, p. 10], and more recent extensions such as DoRA decompose weight updates into magnitude and directional components [27, pp. 3–5], while optimization techniques like LoRA-GA emphasize gradient alignment for faster convergence [52, pp. 4–6]. Comparative and empirical studies, such as the work of Chamdal [6], highlight how LoRA, BitFit, and related methods perform differently depending on task requirements, while broader analyses identify systematic strengths and weaknesses of PEFT approaches across data scales and task types, providing practical guidance for method selection [43]. At the same time, theoretical research offers deeper explanations of why these methods succeed in practice, with optimization landscape analyses clarifying their convergence properties [26] and investigations into representational capacity showing that although PEFT methods are efficient, they remain a constrained subset of full fine-tuning capabilities [29, p. 9]. Finally, complementary work on data selection for task-specific fine-tuning underscores that efficiency can be further enhanced by curating training data appropriately, thus extending the impact of PEFT beyond parameter reduction alone [30, p. 10].

1.4 Scope and Limitations

This study is limited in scope due to the constraints of a Bachelor thesis, particularly regarding time, but also computational resources, and data availability. The focus is restricted to fine-tuning a single LLM with LoRA, without exploring alternative architectures or LoRA derivatives (as described in Section 1.3.2). The training and evaluation rely on an AI-generated dataset outside the authors' domain expertise, which could not undergo extensive validation. These factors constrain the generalizability of the results but still provide insights into the applicability of LoRA-based domain-specific knowledge injection under resource limitations.

As in all machine learning applications, both dataset quality and hyperparameters strongly influence outcomes, yet extensive tuning was not feasible. A further methodological limitation arises from using a Large Language Model for evaluation: although scalable and efficient, such evaluation is inherently non-deterministic, risks bias when architectures overlap, and challenges reproducibility due to sensitivity to prompt design and parameter settings. To mitigate these issues, the generation temperature was fixed at 0, but results should still be interpreted with caution.

2 Methodology

This section outlines the research design, data collection, tools and instruments used, evaluation strategy, and ethical considerations adopted in this study to investigate the effectiveness and efficiency of LoRA-based fine-tuning for domain-specific knowledge injection in LLMs.

2.1 Research Design

This study employs an experimental research design to evaluate the effectiveness of LoRA for fine-tuning LLMs. Three conditions are compared: a non-fine-tuned baseline, a LoRA-fine-tuned model trained on a domain-specific corpus in academic and career counseling, and a RAG pipeline as an alternative approach to domain knowledge integration. The independent variable is the fine-tuning strategy (none, LoRA, or RAG), while the dependent variables comprise performance (answer relevancy, faithfulness, hallucination reduction, factual correctness, and general benchmarks) and efficiency (training time, inference latency, and VRAM usage).

Performance is assessed through traditional metrics (BLEU [38], ROUGE, BERTScore [57]) and LLM-as-a-judge evaluations, whereas efficiency is measured via resource profiling during training, inference, and dataset preparation. Although potential threats to validity exist, such as biases from synthetic data and the non-determinism of LLM-based evaluation, these are mitigated by preprocessing, fixed evaluation parameters, and complementary metrics. Overall, the design enables a controlled comparison between parameter-efficient fine-tuning and retrieval-based approaches, thereby providing a systematic framework to analyse trade-offs in domain-specific performance and computational efficiency.

2.2 Dataset Creation

To construct a domain-specific corpus for fine-tuning language models in academic and career counseling, multiple data sources were combined. The primary corpus consisted of open-access academic literature, provided by a faculty advisor, and was processed through two parallel pipelines to ensure quality. The first relied on a custom text extraction script, which required extensive manual verification due to formatting inconsistencies. The second used the open-source `meta-synthetic-data-kit` to generate structured question-answer (QA) pairs, with documents split manually into coherent chapters. Because a sliding-window approach allowed for longer chunks, the latter produced fewer but more comprehensive QA pairs, and was therefore smaller overall. Due to time constraints, only the better-performing dataset was used in evaluation.

Both datasets were preprocessed with the same toolkit and tailored prompt templates, designed to produce pedagogically relevant QA pairs. The QA generation employed `Llama-3.3-70B-Instruct`, selected over commercial APIs for cost efficiency. A reasoning model was not used due to runtime limitations.

For evaluation, an independent test set was created using `GPT-4.1-mini` via the OpenAI API, chosen for its reliability in instruction following, while being cost efficient [34]. This also reduced the risk of overlap between training and evaluation data.

2.3 Training Parameters

The training process involved fine-tuning the selected language model on the curated dataset with a per-device batch size of 2 and gradient accumulation over 4 steps. Training was performed for 3 epochs with 10 warm-up steps using the `adamw_8bit` optimizer at a learning rate of 1×10^{-4} and `bf16` precision, while reproducibility was ensured through a fixed random seed of 3407. LoRA fine-tuning was applied without dropout or bias terms, with gradient checkpointing enabled via `unsloth` [49]. The sequence length was set to 4096 tokens, and LoRA adapters were applied to attention and MLP modules (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`). Only configurations where

$\alpha = r$ or $\alpha = 2r$ were tested. These parameters are either optimized within `unsloth` for efficiency or recommended in its LoRA hyperparameter guide [49, LoRA hyperparameters guide].

2.4 Tools and Instruments Used

The study employed a range of tools and libraries to facilitate fine-tuning and evaluation. The hardware setup relied on a high-performance computing (HPC) cluster with NVIDIA GPUs, primarily using the `A100 SXM4 80GB` for evaluation and the `A100 SXM4 40GB` for training, as the latter provided sufficient memory and reduced the risk of out-of-memory errors.

For dataset creation, the `meta-synthetic-data-kit` was used to generate question–answer pairs from open-access literature, as described in Section 2.2.

Fine-tuning was conducted with `Unsloth`, which offers training, inference, and evaluation up to "2× faster with 70% less VRAM" [49]. To further reduce memory and computational costs, quantization was applied for k-bit quantization.

Model selection was constrained to small (≤14B parameters), non-MoE, permissively licensed, German-capable LLMs. Candidates included `phi-4` (14.7B), `Gemma 3` (1B, 4B, 12B), and `Llama-3.1-8B`, with the latter chosen due to the availability of an official "Instruct" model by `Unsloth`. The `LLM Explorer` platform supported the model selection [31].

Evaluation relied on `deepeval` for LLM-as-a-judge metrics, chosen for its flexibility compared to alternatives such as `phoenix`, `ragas`, or `opik`. General-purpose benchmarks were conducted with the `lm evaluation harness` [4], using MMLU [15], SuperGLUE [51], and TruthfulQA [25]. This was complemented with custom metrics for domain-specific evaluation, as prior work suggests knowledge injection via LoRA adapters can reduce general performance [42, p. 9].

Finally, (hyper-)parameters followed the Unsloth documentation and the LoRA hyperparameters guide [49, LoRA hyperparameters guide].

2.5 Evaluation Criteria

This section will outline the evaluation criteria employed in this study, detailing the metrics and why they were chosen. Since there is currently no standardized evaluation framework established specifically for this dataset, it is crucial to define clear evaluation criteria to enable meaningful comparisons.

2.5.1 Performance Metrics

This study employs a range of performance metrics to evaluate the effectiveness of the selected model. These metrics are designed to capture various aspects of model performance, including accuracy, factuality, and relevance.

Traditional Metrics

BLEU, introduced by Papineni et al. [38], was one of the earliest metrics for machine translation and remains widely used. It measures exact n-gram overlap between generated and reference text [38, pp. 4–5]. While not capturing semantic similarity, higher BLEU scores still indicate lexical correspondence. In this study, BLEU-1 to BLEU-4 are reported,

with a smoothing method (adding a constant $\epsilon = 0.1$ to the numerator if required) to mitigate the problem of zero values [5, translate.bleu_score.method1].
ROUGE, proposed by Lin [24], also relies on overlap but focuses on content coverage. Variants such as ROUGE-1, ROUGE-2, and ROUGE-L are included to capture both lexical and structural similarity [24, pp. 74–76], while ROUGE-W is excluded due to its disproportionate emphasis on consecutive matches.
BERTScore [57] is used for its context-aware semantic evaluation, complementing BLEU and ROUGE. All three components—recall, precision, and F1—are reported [57, p. 4].

LLM-as-a-judge metrics

These metrics can be categorized into two types: custom metrics and generic metrics. The generic metrics employed in this study are task-specific RAG evaluation metrics, chosen because the task is knowledge-intensive. The custom metric, derived from G-Eval [9, G-Eval], is referred to as **Factual Correctness**. This metric takes the input, the output, and the provided context as parameters, and applies the criterion: "`Determine whether the actual output is factually correct based on the given context`".

Conceptually, Factual Correctness is designed to serve a role analogous to the F_1 score in classification tasks, as it integrates two complementary aspects: avoiding unsupported statements and ensuring the complete and accurate inclusion of relevant facts from the context [44, p. 114]. In this analogy, the **Hallucination** metric is comparable to Precision, measuring whether the output contains only information supported by the context [9, Hallucination]. Conversely, the **Faithfulness** metric parallels Recall, assessing the degree to which the output captures all pertinent information from the retrieval context without omission [9, Faithfulness]. Just as the F_1 score synthesises Precision and Recall into a single value, Factual Correctness is set up to combine the considerations of Hallucination (Precision) and Faithfulness (Recall) into an overall measure of truthfulness relative to the context. This analogy was purposefully chosen to facilitate interpretability and comparison by mirroring a widely understood evaluation paradigm.

For RAG systems, one additional generic metric is employed. The **Contextual Relevancy** metric evaluates the degree to which the retrieved context is pertinent to the input question [9, Contextual Relevancy]; without sufficiently relevant context, factual correctness is improbable.

All models are evaluated using the **Answer Relevancy** metric, which measures the degree to which the generated output directly responds to the input question [9, Answer Relevancy].
The number of metrics is chosen as the developer of the framework states, that 1-2 custom metrics and 2-3 generic metrics are best [17, Choosing your evaluation metrics]. Computation formulas for the generic metrics are provided in the Appendix-Section B.

General capabilities benchmarks

MMLU, SuperGLUE, and TruthfulQA are used to evaluate general model capabilities, while custom metrics assess domain-specific knowledge injection. This distinction is necessary as LoRA-based adaptation can reduce general LLM performance [42, p. 9].

MMLU measures reasoning across diverse academic domains, serving as a proxy for general-purpose knowledge [15, pp. 4–5]. **SuperGLUE** evaluates natural language understanding and commonsense reasoning [51, pp. 5–6], while **TruthfulQA** targets factuality and resistance to plausible but false information [25, pp. 4–5]. Generative TruthfulQA was excluded, as its open-ended format complicates controlled comparison. Collectively, these benchmarks provide rigorous criteria to detect performance trade-offs and balance general with domain-specific capabilities.

2.5.2 Efficiency Metrics

For the efficiency metrics, both **time** and **VRAM usage** were captured, as these represent the primary constraints in large model training and deployment. VRAM is the critical resource, as it limits model size, batch size, and sequence length. Quantization strategies were considered, since they reduce memory consumption without substantially affecting accuracy. VRAM usage was measured with `PyTorch` [39] profiling and `Unsloth` monitoring tools across all stages of the pipeline, including training, inference, and dataset creation.

Temporal efficiency was evaluated by recording training time, dataset creation, implementation overhead, and inference latency. These measurements reflect both the computational cost of optimization and the practical feasibility of large-scale deployment, where low-latency predictions and reduced development time are crucial.

2.6 RAG Pipeline

For contextual comparison, the models are evaluated against a RAG pipeline, implemented with the `Haystack` framework by deepset [10]. The pipeline begins with the `SentenceTransformersTextEmbedder` [10, SentenceTransformersTextEmbedder], which encodes the query into a dense vector using `all-MiniLM-L6-v2`, chosen for its compact size (22MB) and accuracy [48]. The resulting embedding is searched against the document store by the `InMemoryEmbeddingRetriever` [10, InMemoryEmbeddingRetriever], with the number of candidates (`top_k=5`) set to 3·`top_k` to ensure the correct context is retrieved. These candidates are reranked by the `SentenceTransformersSimilarityRanker` [10, SentenceTransformersSimilarityRanker], which applies the cross-encoder `ms-marco-MiniLM-L-6-v2` [8] for efficient relevance refinement, reducing the set to `top_k` passages. The `PromptBuilder` [10, PromptBuilder] then integrates these passages with the query into a structured prompt using a custom `Jinja2` [37] template. Finally, the `HuggingFaceLocalGenerator` [10, HuggingFaceLocalGenerator], based on the instruction-tuned `unsloth/Meta-Llama-3.1-8B-Instruct`, produces the final output under controlled decoding parameters such as maximum length and sampling temperature. This setup ensures retrieval-grounded yet fluent and adaptable answers.

2.7 Data Analysis Techniques

For the analysis of experimental results, both computational and human-in-the-loop methods were applied to ensure reliability and comprehensiveness. Numerical analysis was conducted with `NumPy` [14] for efficient statistical operations and `Pandas` [46] for flexible data manipulation, with pivot tables facilitating structured comparisons across datasets, metrics, and model configurations. To investigate trends and anomalies, visualizations were generated using `Matplotlib` [47] and `Seaborn` [53], enhancing the interpretability of statistical outcomes. Finally, random samples of model outputs were manually reviewed to validate automated analyses and assess qualitative aspects beyond numerical metrics, thereby strengthening the robustness and credibility of the study.

2.8 Ethical Considerations

An important ethical consideration of this study is the commitment to **transparency and reproducibility**. All experiments, including training, inference, and dataset creation, were carried out using the open-source tools mentioned in Section 2.4 and Section 2.7. This choice enables independent verification, fosters reuse and extension of methods, and strengthens the integrity of scientific research.

Another central concern is the **environmental impact** of large-scale machine learning experiments, as training, dataset construction, and inference consume significant energy and generate a carbon footprint. To address this, efficiency techniques such as quantization and resource optimization were employed, reducing memory and computational demands without major performance loss. These measures support the call for more sustainable approaches in artificial intelligence research.

3 Results

This section presents the results of the study, including the performance and efficiency metrics of the fine-tuned LLM using LoRA for domain-specific knowledge injection. As outlined in Section 2.2, only the more promising dataset is subjected to further evaluation. Initial experiments with three fine-tuned models, the base model, and the RAG pipeline revealed that the dataset created using the `meta-synthetic-data-kit` consistently yielded superior results. This finding is illustrated in the following Table 1.

Models	Answer relevancy	Faithfulness	Hallucination	Factual correctness	Contextual relevancy
txt					
r16-a16	0.956	0.866	0.468	0.621	-
r32-a64	0.960	0.854	0.462	0.604	-
r128-a128	0.960	0.869	0.472	0.613	-
RAG	0.863	0.959	0.371	0.785	**0.716**
Base	0.727	0.888	0.754	0.566	-
pdf					
r16-a16	0.951	0.883	0.435	*0.721*	-
r32-a64	***0.968***	0.882	0.478	0.712	-
r128-a128	0.948	*0.876*	*0.431*	0.688	-
RAG	0.886	**0.965**	**0.314**	**0.836**	0.670
Base	0.754	0.877	0.839	0.696	-

Table 1: Quality-oriented mean performance comparison for dataset selection

The models in the `txt` section were fine-tuned on the dataset that originated from the custom script, while those in the `pdf` section were trained on the dataset generated using the whole pipeline of the `meta-synthetic-data-kit`. All reported values represent mean scores across the full evaluation set, comprising 1516 samples for `txt` and 255 samples for `pdf`. The Hallucination metric exhibits a binary behavior, which is likely a consequence of the limited number of claims per instance; similarly, Faithfulness often approaches binary values, indicating that many outputs contained very few statements, which is logical given concise QA pairs. The prefixes `r` and `a` denote the LoRA rank parameter and scaling factor α, respectively. Contextual relevancy is reported exclusively for retrieval-based models, as it is not applicable to fine-tuned models. **Bold** values highlight the overall best-performing model across all categories, whereas *italic* values mark the strongest fine-tuned LoRA variant. For Hallucination, lower values correspond to better outcomes, while for all other metrics higher values are preferred.

Overall, the models trained on the `pdf` dataset consistently outperform those trained on the `txt` dataset across all metrics, with the exception of Contextual Relevancy, where the latter achieves the higher score. In addition to yielding higher mean values, the `pdf` models also demonstrate a smaller variance in performance, further supporting their reliability. Consequently, the `pdf` dataset is selected for all subsequent experiments. Supplementary results for additional metrics, such as ROUGE, BLEU, and BERTScore, as well as the general capability benchmarks, are presented in Appendix-Section C.1. However, these were not decisive for dataset selection, since the primary focus of this study is maximizing domain-specific knowledge injection efficiency. Accordingly, the metrics reported in Table 1 are prioritized for the dataset choice.

3.1 Performance Metrics

The results of the performance metrics are presented in the following subsections, divided into the LLM-as-a-judge metrics, traditional metrics, and general capability benchmarks.

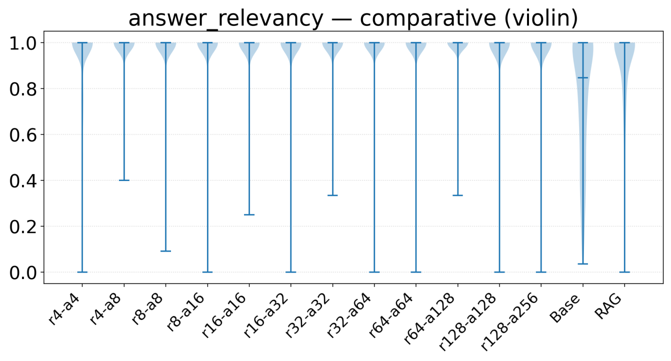
3.1.1 LLM-as-a-Judge Evaluation

The following table presents a comprehensive overview of the model performance across the LLM-as-a-judge metrics.

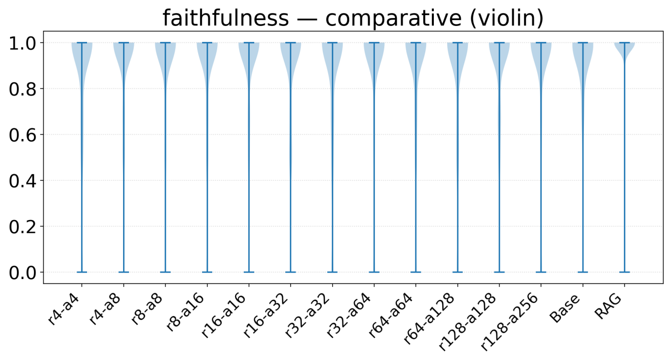
Models	Answer relevancy	Faithfulness	Hallucination	Factual correctness
r4-a4	0.936	0.858	0.471	0.700
r4-a8	0.962	0.890	0.459	0.739
r8-a8	0.960	0.887	0.451	0.666
r8-a16	0.965	0.878	0.486	0.665
r16-a16	0.951	0.883	0.435	0.721
r16-a32	0.958	0.878	0.463	0.703
r32-a32	0.963	0.850	0.498	0.681
r32-a64	0.968	0.882	0.478	0.712
r64-a64	0.964	0.870	0.463	0.708
r64-a128	***0.977***	0.859	0.498	*0.747*
r128-a128	0.948	0.876	*0.431*	0.688
r128-a256	0.953	*0.889*	0.482	0.700
RAG	0.886	**0.965**	**0.314**	**0.836**
Base	0.754	0.877	0.839	0.696

Table 2: Model Performance on Quality Metrics

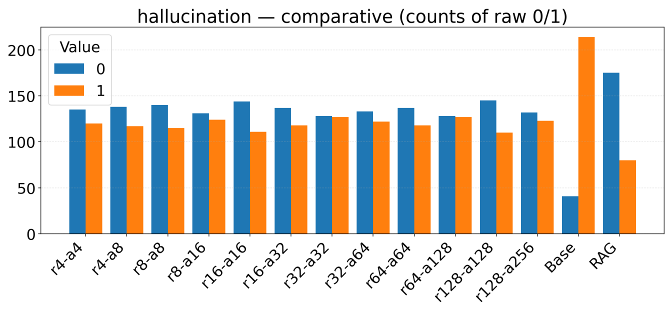
The RAG model consistently outperforms the LoRA models across all metrics except for answer relevancy, where the LoRA models demonstrate substantially better results, achieving nearly 10 percentage points (%pt) higher performance in the best-performing configuration. Nevertheless, RAG still surpasses the base model with an uplift of approximately 13%pt. In the remaining metrics, RAG holds the advantage over LoRA. Specifically, in Faithfulness, RAG is 6.6%pt better than the strongest LoRA model, while the LoRA models themselves show no significant improvement over the base model. In Hallucination, RAG achieves slightly more than 12%pt higher performance, though the LoRA models still provide a substantial uplift compared to the base model, with the best achieving a reduction of approximately 40%pt in hallucinations on average. Regarding Factual Correctness, RAG also dominates, outperforming by around 9%pt. The LoRA models, by contrast, only show modest gains over the base model in this metric, with the best result being an improvement of about 5%pt, though performance fluctuates considerably, with some models even performing worse than the baseline. It is further noteworthy that the best-performing LoRA configurations were consistently trained with a rank ≥ 64. While these findings are encouraging and provide a clear comparative overview, the fluctuations in model performance across metrics must be taken into account. The following figures illustrate this variability in greater detail.



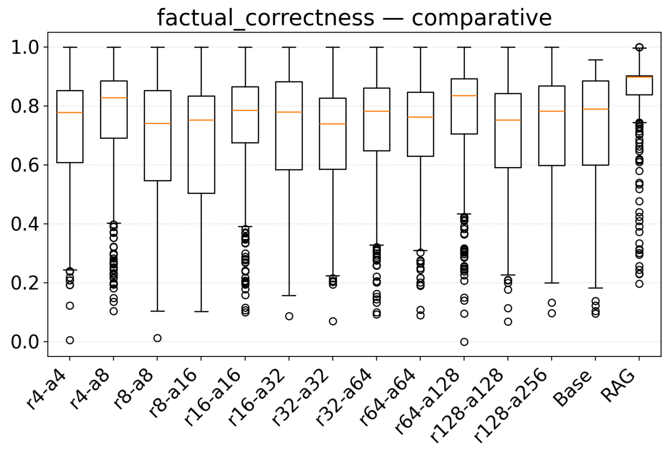
(a) Answer Relevancy



(b) Faithfulness



(c) Hallucination



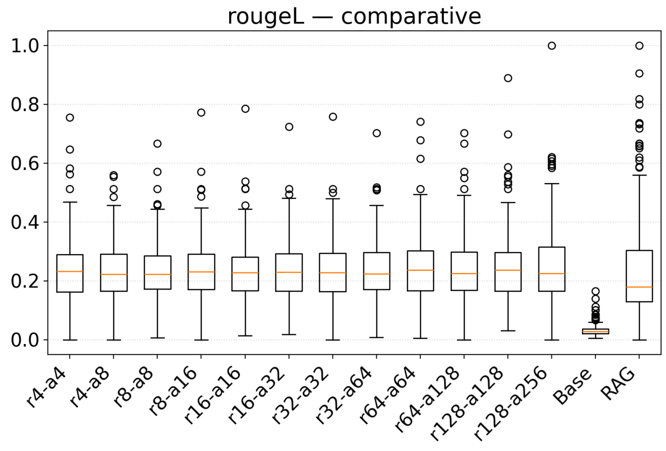
(d) Factual Correctness

Figure 1: Comparison of model performance with respect to variability

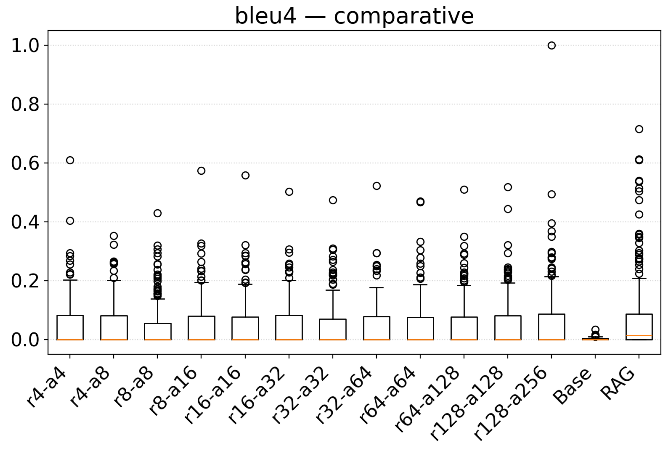
Violin plots are used where boxplots didn't show the distribution well due to very "skewed" data. The variability in model performance across different metrics highlights that the mean performance from Table 2 can be misleading, as the mean does not show the distribution of performance. It is noticeable that the RAG pipeline consistently outperforms all other models across all metrics apart from the Answer Relevancy—even with a lower variability in most cases. Especially Hallucination shows a superior performance for the RAG model, but LoRA models still provide a big performance boost compared to the base model. In both Faithfulness and Factual Correctness RAG shows a more consistent performance than the LoRA models. In these figures no indication between higher ranks and better performance can be found as all models show a high variability in performance.

3.1.2 Traditional Metrics

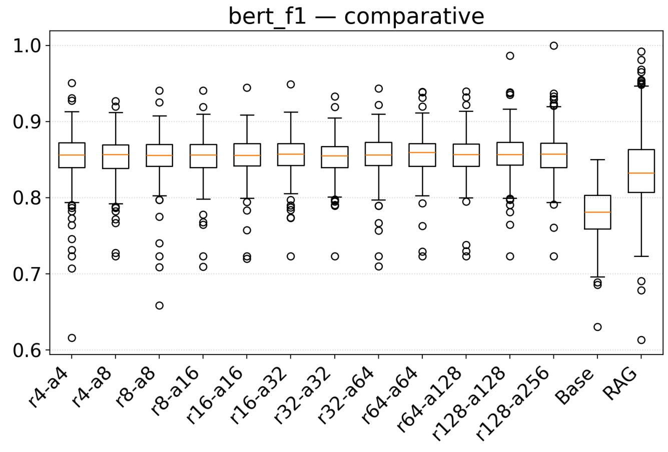
For the traditional metrics the boxplots will directly provide a better overview of model performance:



(a) ROUGE-L (1 ≈ 0.1 better, 2 ≈ 0.1 worse, see Figure 6)



(b) BLEU 4 (1, 2, 3 very similar, see Figure 7)



(c) BERTScore F1 (Precision and Recall nearly the same, see Figure 8)

Figure 2: Boxplots of model performance on traditional metrics

ROUGE and BLEU show scores that do not go along with the previous quality metrics. Still, in ROUGE LoRA consistently outperform the RAG model by a small amount (≈ 3.5%pt) and the base model by a huge difference as the base model has a score near 1%. BLEU shows an even worse performance but LoRA and RAG have a very similar performance, while the base model has a near zero performance. The BERT-F1 Score correlates better with the LLM-as-a-judge metrics, having a mean score near 85.5% for LoRA models and approximately 83.7% for the RAG model. The variability of the RAG model is also higher than for the LoRA models. The base model is beaten by both with a mean score at around 77.9%.

3.1.3 General Capability Benchmarks

The RAG model is not included in the following general capabilities benchmarks as the prompt used in the pipeline specifies that the model should generate "I cannot find the answer in the provided context." if the question is not answerable from the provided context. But the base model, which is the same model as used in the RAG pipeline, is included to provide a better comparison.

MMLU

Model	Humanities	Other	Social Sciences	STEM
r4-a4	0.589	0.698	0.736	0.544
r4-a8	0.590	0.701	0.737	0.548
r8-a8	0.587	***0.702***	*0.738*	0.550
r8-a16	0.588	0.696	0.737	0.545
r16-a16	0.589	0.697	0.736	*0.550*
r16-a32	0.593	0.692	0.732	0.543
r32-a32	***0.596***	0.695	*0.738*	0.549
r32-a64	0.583	0.694	0.735	0.547
r64-a64	0.572	0.694	0.735	0.545
r64-a128	0.556	0.690	0.728	0.542
r128-a128	0.557	0.690	0.730	0.536
r128-a256	0.545	0.671	0.704	0.529
Base	0.586	0.696	**0.749**	**0.551**

Table 3: MMLU Category Breakdown

The results of MMLU show that a higher rank and alpha degrade the model's performance (in this benchmark), while some low(er) ranks and alphas even increase the performance by a tiny bit, but that could also be fluctuations in the generation process since the difference is $\leq 1\%pt$.

SuperGLUE

Model	boolq_acc	cb_acc	cb_f1	copa_acc	multirc_acc	record_em	record_f1	rte_acc	wic_acc	wsc_acc
r4-a4	***0.863***	0.875	***0.854***	0.930	0.265	*0.898*	*0.906*	0.809	***0.593***	0.750
r4-a8	0.862	0.857	0.775	0.930	0.259	0.896	0.904	***0.812***	0.583	0.740
r8-a8	0.855	***0.893***	0.851	0.930	0.255	0.896	0.904	0.809	0.574	0.750
r8-a16	0.855	***0.893***	0.851	0.930	0.252	0.894	0.902	0.809	0.571	*0.760*
r16-a16	0.855	0.857	0.772	0.940	0.245	0.894	0.902	0.805	0.582	0.731
r16-a32	0.850	0.875	0.786	0.930	0.249	0.890	0.898	0.798	0.575	0.731
r32-a32	0.852	0.839	0.689	0.940	0.250	0.890	0.898	0.798	0.578	0.731
r32-a64	0.848	0.875	0.837	0.930	0.246	0.890	0.898	0.787	***0.593***	0.750
r64-a64	0.850	0.839	0.743	***0.960***	0.257	0.891	0.898	0.794	0.571	0.731
r64-a128	0.850	0.839	0.812	0.930	0.312	0.891	0.898	***0.812***	0.561	0.740
r128-a128	0.843	0.821	0.734	0.930	0.282	0.888	0.896	0.805	0.575	0.731
r128-a256	0.827	0.839	0.673	0.940	***0.327***	0.880	0.887	0.783	0.541	0.740
Base	0.862	0.839	0.810	0.930	0.226	**0.901**	**0.908**	**0.812**	0.575	**0.769**

Table 4: SuperGLUE Performance Comparison

This table shows that while high ranks degrade the performance in some benchmarks (e.g. CommitmentBank (cb) [32] F1-Score), they can also improve the performance in others (e.g. Multi-Sentence Reading Comprehension (multirc) [20]).

TruthfulQA

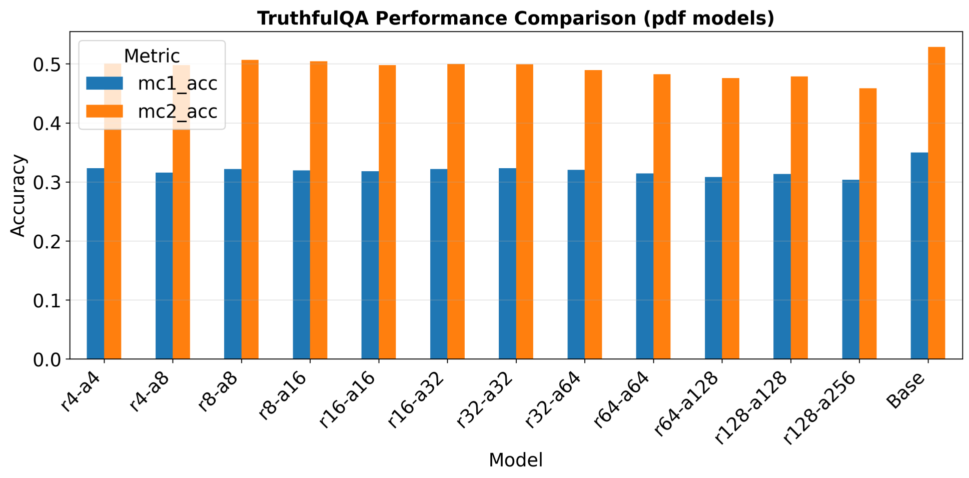


Figure 3: TruthfulQA Performance Comparison

The results of TruthfulQA show again that a higher rank degrades performance in respect to the base model. Notably, the LoRA model with only a rank of 4 already has an ≈ 3%pt lower performance in this task. It is also apparent that the multiple-choice questions with only one correct answer (mc1) are degrading more than the ones with multiple correct answers (mc2).

3.2 Efficiency Metrics

The efficiency metrics include the training and generation times, as well as VRAM usage during these phases, which is shown in the following figures. Additionally, the dataset creation and implementation is considered in this section.

3.2.1 Temporal Efficiency

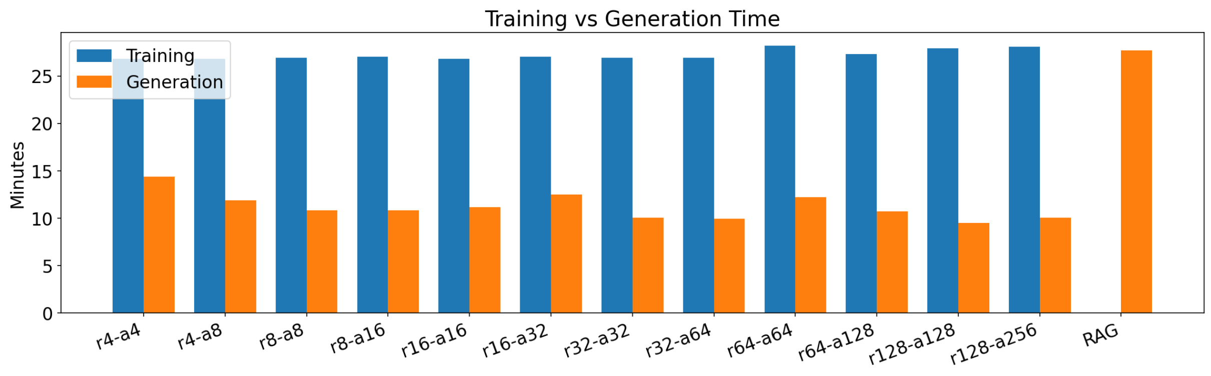


Figure 4: Training and Generation Times

The training for the LoRA models took a nearly constant time of around 27 minutes. The RAG model does not have to be trained, but the time is used for embedding the context, which is significantly lower than the training time of the LoRA models needing only 0.2 minutes. Generation/inference times for the LoRA models is about half of the

RAG model, ranging from 10 to 14 minutes against nearly 27.7 minutes for the 255 questions equalling ≈ 2.35-3.3 seconds per question for LoRA models and ≈ 6.35 seconds per question for the RAG model.

The time it takes for the dataset creation process is totally dependent on the model, quantization, batch size, corpus, and other factors, making it difficult to provide a one-size-fits-all estimate. In this study with close to 1200 pages of text, it took around 18 hours.

The implementation of the LoRA script required approximately 30 minutes, primarily due to more complex data formats and the additional hyperparameters involved in fine-tuning the model. In contrast, the RAG script was completed in about 15 minutes, facilitated by the extensive availability of implementation examples, although the required time strongly depends on the complexity of the specific pipeline.

3.2.2 VRAM Usage

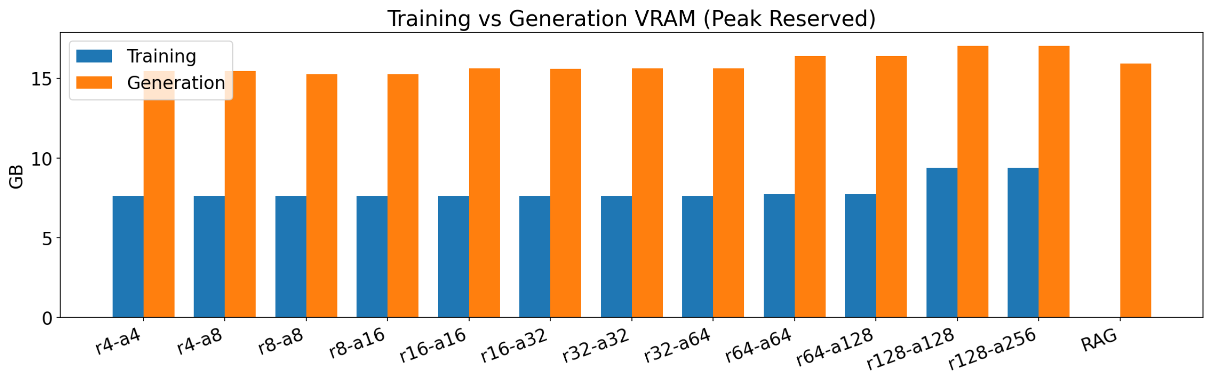


Figure 5: VRAM Usage during Training and Generation

The reserved VRAM for the LoRA models during training is ≈ 7.6 GB, with only minor increases for higher ranks, such as an additional 120 MB at rank 64, while rank 128 peaks at 9.6 GB. In contrast, the RAG pipeline consistently requires substantially more memory, with ≈ 15.9 GB consumed across embedding and generation components. This exceeds the VRAM usage of all LoRA configurations, where rank 4 occupies ≈ 15.4 GB and shows a steady increase up to ≈ 15.6 GB at rank 32, while rank 64 rises to ≈ 16.3 GB and rank 128 approaches 17 GB.

The VRAM usage during the dataset creation process is—again—highly dependent on the model architecture, quantization strategy, batch size, and several other factors, which makes it challenging to provide a universally applicable estimate. In this study, however, the process required approximately 160 GB of VRAM.

4 Discussion

The study highlights the strengths and weaknesses of both LoRA and RAG approaches in fine-tuning large language models. While LoRA excels in efficiency and task-specific alignment, RAG demonstrates superior factual accuracy and contextual understanding.

4.1 Interpretation of Results

The results indicate that LoRA can successfully inject domain-specific knowledge, as performance improves compared to the base model. However, it underperforms relative to RAG in factuality and hallucination, while showing advantages in answer relevancy. This suggests that fine-tuning with LoRA helps the model align more closely with domain-specific contexts. Higher ranks ($r \geq 64$) generally improve quality metrics in LLM-as-a-judge evaluations, but they simultaneously degrade performance on general capabilities benchmarks, pointing to overfitting or catastrophic forgetting. Nevertheless, the fact that LoRA models occasionally even surpass the base model on certain general benchmarks indicates a complex interplay between fine-tuning data, latent model knowledge, and evaluation tasks; as the benchmark tasks do not appear to resemble the fine-tuning data, no clear direction of this effect can be inferred. Moreover, higher rank and scaling factor combinations do not always yield better results, which may stem from collapsing gradients caused by the conventional scaling $\frac{\alpha}{r}$ [18, pp. 3–5]. In contrast, RAG achieves more stable results, likely because it leverages the pre-trained base model and augments it with retrieved external knowledge.

In terms of efficiency, LoRA demonstrates its strength as a parameter-efficient fine-tuning method. Training requires only about 30 minutes on a single GPU and does not affect inference speed, while VRAM usage rises with rank. RAG, in turn, avoids fine-tuning but requires embedding the documents; in this study, embedding time was just 0.2 minutes. Thus, although LoRA is viable in resource-constrained environments and surpasses the base model, RAG provides stronger performance with minimal preprocessing overhead.

A key limitation for LoRA is dataset construction, as performance depends heavily on the quality and representativeness of the training data, which increases costs and time requirements. By comparison, RAG pipelines are simpler, relying largely on extraction and embedding. Finally, traditional metrics such as ROUGE and BLEU prove inadequate for this use case, while BERTScore correlates better with LLM-as-a-judge but still fails to reliably capture factual correctness, which is critical for evaluating knowledge injection.

4.2 Comparison with Existing Literature

Existing research on knowledge injection in LLMs largely aligns with our findings, highlighting a trade-off between parameter-efficient fine-tuning methods such as LoRA and RAG. In general, RAG is widely reported to improve factual accuracy and reduce hallucinations by grounding responses in external knowledge. For instance, more recent domain-specific studies like conducted by Pingua et al. [41] consistently show that RAG-based and hybrid (fine-tuning *and* retrieval) models outperform fine-tuning alone [41, p. 23]. Our observation that RAG achieved higher factuality than LoRA mirrors these trends, though we also found that LoRA surpassed RAG in terms of answer relevancy, highlighting the context-dependent nature of these approaches.

That said, fine-tuning approaches like LoRA provide advantages in tailoring models to domain-specific contexts. By updating weights in a parameter-efficient manner, LoRA can align the model's style and focus with in-domain data, yielding more contextually coherent and relevant answers. Prior studies have shown that LoRA-adapted models preserve much of the base model's behavior while integrating new domain knowledge, making them less disruptive than full fine-tuning [3, p. 13]. However, our finding that higher LoRA ranks ($r \geq 64$) degraded general-task performance is consistent with established results on catastrophic forgetting [54, p. 1], [42, p. 1], [55, p. 1]. Kalajdzievski [18] provides a theoretical basis: the conventional scaling factor $\frac{\alpha}{r}$ can lead to diminishing returns and unstable gradients at high ranks [18, pp. 3–5], which explains why performance did not improve monotonically in our experiments. Recent work on adaptive rank allocation, such as ARD-LoRA [45], confirms that learnable scaling strategies can mitigate these issues and achieve near full fine-tuning performance with minimal parameters [45, p. 8].

In contrast, RAG inherently avoids modifying model parameters, leading to more stable retention of general capabilities. Theoretical reviews describe RAG as providing a form of "live memory" [33, p. 6], where retrieval modules supply factual grounding without altering internal weights. This modular design makes RAG particularly effective for incorporating new information on the fly, especially when compared with fine-tuning for knowledge injection [35, p. 8]. Recent advances in graph-based retrieval [2] further strengthen RAG's role in reducing hallucinations, with Fact-RAG achieving faithfulness scores as high as 0.97 [2, p. 59].

Efficiency considerations also support our findings. LoRA is widely recognized for requiring orders-of-magnitude fewer trainable parameters and negligible inference overhead [12]. Our reported training time of about 30 minutes on a single GPU is consistent with this efficiency, though it comes at the cost of dataset preparation and quality control, a limitation also highlighted by Paul [40]. By comparison, RAG requires minimal preprocessing—embedding in our case took only 0.2 minutes—and supports rapid updates by refreshing the knowledge index.

Finally, our results echo ongoing concerns about evaluation metrics. Traditional measures like ROUGE and BLEU often fail to capture factual correctness, and while BERTScore correlates more strongly with human judgments [57, p. 9], even it cannot reliably detect hallucinations. Recent surveys as conducted by Li et al. [22] confirm that LLM-as-a-judge approaches improve scalability and interpretability but still suffer from prompt sensitivity, bias, hallucination, and knowledge gaps as well as recency [22, pp. 36–44]. Our findings reinforce the growing consensus that no single metric adequately captures both semantic and factual correctness, making robust evaluation an open challenge.

In summary, our study is consistent with the broader literature: LoRA is efficient and effective for domain alignment but can suffer from overfitting and forgetting at high ranks, while RAG provides more stable factual grounding by leveraging external knowledge.

4.3 Implications of Findings

The findings of this study carry important implications for both research and practice in domain-specific applications of LLMs. First, the trade-off observed between LoRA and RAG suggests that the choice of knowledge injection method should be guided by the intended application context. LoRA is well-suited for scenarios where relevance and contextual alignment are paramount, as it allows the model to adapt closely to domain-specific language and style. However, its vulnerability to catastrophic forgetting, overfitting at higher ranks, and vanishing gradients limits its utility in settings where general-purpose performance and factual consistency are equally important.

Conversely, RAG demonstrates greater stability and factual grounding by leveraging external retrieval without altering model parameters. This makes it a preferable choice in contexts where factual accuracy and knowledge freshness are critical, particularly in fast-evolving fields where external resources can be frequently updated. The modularity of RAG thus positions it as a more robust strategy for applications requiring dynamic adaptation, such as academic counseling or career guidance, where accuracy and trustworthiness outweigh stylistic alignment.

Efficiency considerations further refine these implications. While LoRA offers rapid fine-tuning and parameter efficiency, it incurs significant costs in dataset construction and quality control, which may be prohibitive in resource-constrained environments. By contrast, RAG requires minimal preprocessing and can be deployed more rapidly, underscoring its practicality for organizations with limited computational or data-curation capacity.

Finally, the inadequacy of traditional automatic metrics emphasizes the need for improved evaluation frameworks. The limited reliability of BLEU, ROUGE, and even BERTScore in capturing factual correctness highlights the necessity of adopting LLM-as-a-judge methods or hybrid human-in-the-loop evaluation pipelines in future work. This implies that stakeholders aiming to deploy domain-adapted LLMs must not only choose an appropriate adaptation strategy but also carefully consider evaluation methodologies to ensure robustness and trustworthiness of model outputs.

Taken together, these findings imply that LoRA and RAG should not be viewed as mutually exclusive but rather as complementary strategies, where the choice depends on the balance between relevance, factual accuracy, efficiency, and evaluation reliability demanded by the application domain.

4.4 Limitations of the Study

Due to strict time constraints, the study was confined to a single base model and optimizer, without extensive hyperparameter tuning. Only one base model is especially suboptimal with no possibility of verifying whether the model's pre-training data aligned with the fine-tuning data. Furthermore, the dataset was not drawn from a standardized benchmark but instead generated using AI, which may have introduced biases and limits the overall generalizability of the results. The evaluation also relied primarily on LLM-as-a-judge methods which, while highly scalable, can themselves introduce biases and inconsistencies. Moreover, the study did not investigate hybrid approaches that integrate LoRA with RAG, despite their potential to produce synergistic improvements. The RAG implementation employed in this work was comparatively simple, as its inclusion was not foreseen in the initial research plan; thus, more advanced retrieval strategies could further improve outcomes. Finally, the inherent randomness associated with training and deploying LLMs may also have influenced the reported results.

5 Conclusion

This section summarizes the key findings, contributions to knowledge, and suggestions for future research based on the study's results and limitations.

5.1 Summary of Key Findings

This study assessed the effectiveness and efficiency of Low-Rank Adaptation for fine-tuning Large Language Models in the context of academic and career counseling. The results demonstrate that LoRA-adapted models consistently outperform the base model across most quality metrics, with notable gains in answer relevancy and a reduction in hallucinations. At the same time, Retrieval-Augmented Generation achieved superior performance in factual correctness, faithfulness, and stability, highlighting its advantage in leveraging external knowledge sources for grounding. From an efficiency perspective, LoRA proved to be highly lightweight, requiring only approximately 30 minutes of training on a single GPU, though these benefits are offset by the costs associated with dataset construction and quality assurance. By comparison, RAG avoids training altogether and instead relies on efficient document embedding, rendering it well-suited for rapid deployment scenarios. The research question therefore can be answered as follows: Low-Rank Adaptation is effective for injecting domain-specific knowledge and enhancing contextual relevance, while Retrieval-Augmented Generation excels in factual accuracy and stability. The choice between the two methods should be informed by the specific requirements of the application, balancing the need for relevance, accuracy, efficiency, and evaluation robustness.

5.2 Contributions to Knowledge

This thesis extends the existing literature on parameter-efficient fine-tuning by offering an empirical comparison of LoRA and RAG in a real-world use case. The findings show that LoRA enables effective domain-specific knowledge injection while maintaining a low computational footprint. In contrast, RAG demonstrates superior factual grounding and stability, though at the cost of slower inference times. The analysis further reveals that the relative advantages of the two approaches are highly context-dependent: LoRA is preferable for stylistic and contextual alignment, whereas RAG is more reliable for factual accuracy and adaptability. Moreover, the evaluation with LLM-as-a-judge highlights the limitations of traditional metrics such as BLEU and ROUGE, emphasizing the necessity of more nuanced evaluation frameworks for domain adaptation tasks.

5.3 Suggestions for Future Research

Future work should investigate hybrid methodologies that integrate the strengths of LoRA and RAG, thereby balancing contextual alignment with factual reliability. Extending the analysis across multiple base models, diverse domains, and additional languages would further enhance the generalizability of the findings. Moreover, advancing evaluation strategies by combining LLM-as-a-judge with more human-in-the-loop assessments could provide a more comprehensive view of both semantic quality and factual accuracy. Finally, promising directions include adaptive rank allocation to mitigate catastrophic forgetting, refined scaling strategies to address gradient instability, and advanced retrieval mechanisms to reduce dependency on dataset quality.

References

[1] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, *Intrinsic dimensionality explains the effectiveness of language model fine-tuning*, 2020. arXiv: `2012.13255 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2012.13255`.

[2] M. Barry et al., "GraphRAG: Leveraging graph-based efficiency to minimize hallucinations in LLM-driven RAG for finance data," in *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)*, G. A. Gesese, H. Sack, H. Paulheim, A. Merono-Penuela, and L. Chen, Eds., Abu Dhabi, UAE: International Committee on Computational Linguistics, Jan. 2025, pp. 54–65. [Online]. Available: `https://aclanthology.org/2025.genaik-1.6/`.

[3] D. Biderman et al., *Lora learns less and forgets less*, 2024. arXiv: `2405.09673 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2405.09673`.

[4] S. Biderman et al., *Lessons from the trenches on reproducible evaluation of language models*, 2024. arXiv: `2405.14782 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/2405.14782`.

[5] S. Bird and E. Loper, "NLTK: The natural language toolkit," in *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 214–217. [Online]. Available: `https://aclanthology.org/P04-3031/`.

[6] H. Chamdal, "Comparing parameter efficient finetuning techniques (peft) using datamodels," M.S. thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, May 2024. [Online]. Available: `https://dspace.mit.edu/handle/1721.1/157345`.

[7] Y. Chen, D. Hazarika, M. Namazifar, Y. Liu, D. Jin, and D. Hakkani-Tur, *Inducer-tuning: Connecting prefix-tuning and adapter-tuning*, 2022. arXiv: `2210.14469 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/2210.14469`.

[8] Dataloop AI, *Ms marco minilm l 6 v2*, `https://dataloop.ai/library/model/cross-encoder_ms-marco-minilm-l-6-v2/`, 2023.

[9] DeepEval, *Deepeval documentation*, Accessed: 2025-08-11, 2025. [Online]. Available: `https://deepeval.com/docs/getting-started`.

[10] deepset, *Haystack documentation*, `https://docs.haystack.deepset.ai/docs/intro`, Accessed: 2025-08-24, 2025.

[11] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, *Qlora: Efficient fine-tuning of quantized llms*, 2023. arXiv: `2305.14314 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2305.14314`.

[12] Emergent Mind, "Parameter-efficient fine-tuning," *Emergent Mind Topics*, Aug. 2025. [Online]. Available: `https://www.emergentmind.com/topics/parameter-efficient-fine-tuning`.

[13] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, *Parameter-efficient fine-tuning for large models: A comprehensive survey*, 2024. arXiv: `2403.14608 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2403.14608`.

[14] C. R. Harris et al., “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[15] D. Hendrycks et al., *Measuring massive multitask language understanding*, 2021. arXiv: 2009.03300 [cs.CY]. [Online]. Available: https://arxiv.org/abs/2009.03300.

[16] E. J. Hu et al., *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2106.09685.

[17] J. Ip, *Llm evaluation metrics: Everything you need for llm evaluation*, https://www.confident-ai.com/blog/llm-evaluation-metrics-everything-you-need-for-llm-evaluation, Accessed: 2025-08-14, Confident AI, 2025.

[18] D. Kalajdzievski, *A rank stabilization scaling factor for fine-tuning with lora*, 2023. arXiv: 2312.03732 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2312.03732.

[19] J. Kaplan et al., *Scaling laws for neural language models*, 2020. arXiv: 2001.08361 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2001.08361.

[20] D. Khashabi, S. Chaturvedi, M. Roth, S. Upadhyay, and D. Roth, “Looking beyond the surface: A challenge set for reading comprehension over multiple sentences,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jul. 2018, pp. 252–262. DOI: 10.18653/v1/N18-1023. [Online]. Available: https://aclanthology.org/N18-1023/.

[21] P. Lewis et al., *Retrieval-augmented generation for knowledge-intensive nlp tasks*, 2021. arXiv: 2005.11401 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2005.11401.

[22] H. Li et al., *Llms-as-judges: A comprehensive survey on llm-based evaluation methods*, 2024. arXiv: 2412.05579 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2412.05579.

[23] X. L. Li and P. Liang, *Prefix-tuning: Optimizing continuous prompts for generation*, 2021. arXiv: 2101.00190 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2101.00190.

[24] C.-Y. Lin, “ROUGE: A package for automatic evaluation of summaries,” in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013/.

[25] S. Lin, J. Hilton, and O. Evans, *Truthfulqa: Measuring how models mimic human falsehoods*, 2022. arXiv: 2109.07958 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2109.07958.

[26] X.-H. Liu, Y. Du, J. Wang, and Y. Yu, “On the optimization landscape of low rank adaptation methods for large language models,” in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=pxclAomHat.

[27] S.-Y. Liu et al., *Dora: Weight-decomposed low-rank adaptation*, 2024. arXiv: 2402.09353 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2402.09353.

[28] X. Liu et al., *P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks*, 2022. arXiv: 2110.07602 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2110.07602.

[29] Y. Liu et al., *Look within or look beyond? a theoretical comparison between parameter-efficient and full fine-tuning*, 2025. arXiv: 2505.22355 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2505.22355.

[30] Z. Liu, A. Karbasi, and T. Rekatsinas, *Tsds: Data selection for task-specific model finetuning*, 2024. arXiv: 2410.11303 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2410.11303.

[31] LLM Explorer Team, *Llm explorer: A curated large language model directory*, https://llm-explorer.com/, Accessed: 2025-08-13, Extractum.io, 2025.

[32] M.-C. de Marneffe, M. Simons, and J. Tonhauser, "The commitmentbank: Investigating projection in naturally occurring discourse," *Proceedings of Sinn und Bedeutung*, vol. 23, no. 2, pp. 107–124, Aug. 2019. DOI: 10.18148/sub/2019.v23i2.601. [Online]. Available: https://ojs.ub.uni-konstanz.de/sub/index.php/sub/article/view/601.

[33] A. J. Oche, A. G. Folashade, T. Ghosal, and A. Biswas, *A systematic review of key retrieval-augmented generation (rag) systems: Progress, gaps, and future directions*, 2025. arXiv: 2507.18910 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2507.18910.

[34] OpenAI, *Introducing gpt-4.1 in the api*, https://openai.com/index/gpt-4-1/, Accessed: 2025-09-09, Apr. 2025.

[35] O. Ovadia, M. Brief, M. Mishaeli, and O. Elisha, *Fine-tuning or retrieval? comparing knowledge injection in llms*, 2024. arXiv: 2312.05934 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2312.05934.

[36] S. Pai, *Designing Large Language Model Applications.* O'Reilly Media, 2025, A Holistic Approach to LLMs, ISBN: 978-1-098-15904-7.

[37] Pallets Project, *Jinja: A very fast and expressive template engine*, https://github.com/pallets/jinja, Version 3.1.6, released March 5, 2025, 2025.

[38] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak, and D. Lin, Eds., Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. [Online]. Available: https://aclanthology.org/P02-1040/.

[39] A. Paszke et al., *Pytorch: An imperative style, high-performance deep learning library*, 2019. arXiv: 1912.01703 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1912.01703.

[40] R. Paul, "Domain-specific llm fine-tuning," *Rohan's Bytes*, Apr. 2025. [Online]. Available: https://www.rohan-paul.com/p/domain-specific-llm-fine-tuning.

[41] B. Pingua et al., “Medical llms: Fine-tuning vs. retrieval-augmented generation,” *Bioengineering*, vol. 12, no. 7, 2025, ISSN: 2306-5354. DOI: `10.3390/bioengineering12070687`. [Online]. Available: `https://www.mdpi.com/2306-5354/12/7/687`.

[42] S. Pletenev et al., *How much knowledge can you pack into a lora adapter without harming llm?* 2025. arXiv: `2502.14502 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/2502.14502`.

[43] G. Pu, A. Jain, J. Yin, and R. Kaplan, *Empirical analysis of the strengths and weaknesses of peft techniques for llms*, 2023. arXiv: `2304.14999 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/2304.14999`.

[44] C. J. van Rijsbergen, *Information Retrieval*, 2nd. London: Butterworths, 1979.

[45] H. U. K. Shinwari and M. Usama, *Ard-lora: Dynamic rank allocation for parameter-efficient fine-tuning of foundation models with heterogeneous adaptation needs*, 2025. arXiv: `2506.18267 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2506.18267`.

[46] T. pandas development team, *Pandas-dev/pandas: Pandas*, version latest, Feb. 2020. DOI: `10.5281/zenodo.3509134`. [Online]. Available: `https://doi.org/10.5281/zenodo.3509134`.

[47] The Matplotlib Development Team, *Matplotlib: Visualization with python*, version v3.10.5, Jul. 2025. DOI: `10.5281/zenodo.16644850`. [Online]. Available: `https://doi.org/10.5281/zenodo.16644850`.

[48] R. Tiwari. “Unlocking the power of sentence embeddings with all-minilm-l6-v2.” Accessed: 2025-08-24, Medium. [Online]. Available: `https://medium.com/@rahultiwari065/unlocking-the-power-of-sentence-embeddings-with-all-minilm-l6-v2-7d6589a5f0aa`.

[49] Unsloth-AI, *Unsloth documentation*, `https://docs.unsloth.ai/`, Accessed: 2025-08-03, 2025.

[50] A. Vaswani et al., *Attention is all you need*, 2023. arXiv: `1706.03762 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/1706.03762`.

[51] A. Wang et al., *Superglue: A stickier benchmark for general-purpose language understanding systems*, 2020. arXiv: `1905.00537 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/1905.00537`.

[52] S. Wang, L. Yu, and J. Li, *Lora-ga: Low-rank adaptation with gradient approximation*, 2024. arXiv: `2407.05000 [cs.LG]`. [Online]. Available: `https://arxiv.org/abs/2407.05000`.

[53] M. L. Waskom, “Seaborn: Statistical data visualization,” *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. DOI: `10.21105/joss.03021`. [Online]. Available: `https://doi.org/10.21105/joss.03021`.

[54] X. Wei, G. Li, and R. Marculescu, *Online-lora: Task-free online continual learning via low rank adaptation*, 2024. arXiv: `2411.05663 [cs.CV]`. [Online]. Available: `https://arxiv.org/abs/2411.05663`.

[55] J. Zhang, J. You, A. Panda, and T. Goldstein, “LoRA without forgetting: Freezing and sparse masking for low-rank adaptation,” in *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference*, 2025. [Online]. Available: `https://openreview.net/forum?id=aG0QYJfz6H`.

[56] Q. Zhang et al., *Adalora: Adaptive budget allocation for parameter-efficient fine-tuning*, 2023. arXiv: `2303.10512 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/2303.10512`.

[57] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, *Bertscore: Evaluating text generation with bert*, 2020. arXiv: `1904.09675 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/1904.09675`.

Appendix

A AI usage example

Instruction:
Write academic and scientific for a paper published in a journal.
The following sentence should always be a logical conclusion of the previous one.
All text for the paper should be as Latex code and use the same commands as the given text.

Prompt (written text by the author):

```
Low-Rank Adaptation (LoRA) proposed by \textcite{
    hu2021loralowrankadaptationlarge} belongs\textemdash{}as described in
     \ref{subsubsec:theoretical-foundations-transformer-architecture}\
    textemdash{}to the family of parameter-efficient methods to fine-tune
     large language models \parencite[p. 1]{
    hu2021loralowrankadaptationlarge}. As the name suggests, not all
    parameters are retrained, but the weights are frozen and a trainable
    matrix with a low rank is inserted. In the original paper, this
    happens in every layer of the transformer architecture \parencite[pp.
     4-5]{hu2021loralowrankadaptationlarge}. More specifically, dense
    layers are trained indirectly by optimizing rank decomposition
    matrices. The hypothesis behind this idea is that the weight changes
    during model adaptation have a low \enquote{intrinsic rank} \
    parencite[p. 2]{hu2021loralowrankadaptationlarge}.\\
```

Output (generated by AI):

```
Low-Rank Adaptation (LoRA), proposed by \textcite{
    hu2021loralowrankadaptationlarge}, belongs\textemdash{}as described
    in \ref{subsubsec:theoretical-foundations-transformer-architecture}\
    textemdash{}to the family of parameter-efficient methods for fine-
    tuning large language models \parencite[p. 1]{
    hu2021loralowrankadaptationlarge}. As the name suggests, not all
    parameters are retrained; instead, the pre-trained weights are frozen
     and a trainable matrix with low rank is inserted into the
    architecture. In the original formulation, this reparametrization is
    applied in every layer of the transformer architecture \parencite[pp.
     4-5]{hu2021loralowrankadaptationlarge}. More specifically, the dense
     layers are indirectly trained by optimizing the factors of a low-
    rank decomposition. This is motivated by the hypothesis that the
    weight updates required during model adaptation exhibit a low \
    enquote{intrinsic rank} \parencite[p. 2]{
    hu2021loralowrankadaptationlarge}.
```

B Metrics Formulas

B.1 Answer Relevancy

$$\text{AnswerRelevancy} = \frac{\text{NumberofRelevantStatements}}{\text{TotalNumberofStatements}} \tag{4}$$

[9, Answer Relevancy]

B.2 Faithfulness

$$\text{Faithfulness} = \frac{\text{NumberofTruthfulClaims}}{\text{TotalNumberofClaims}} \tag{5}$$

[9, Faithfulness]

B.3 Contextual Relevancy

$$\text{ContextualRelevancy} = \frac{\text{NumberofRelevantStatements}}{\text{TotalNumberofStatements}} \tag{6}$$

[9, Contextual Relevancy]

B.4 Hallucination

$$\text{Hallucination} = \frac{\text{NumberofContradictedContexts}}{\text{TotalNumberofContexts}} \tag{7}$$

[9, Hallucination] Due to the low total number of x in the dataset, all metrics tend to be binary or nearly binary.

C Additional Evaluation Results

C.1 Dataset Selection

Model	**bleu1**	**bleu2**	**bleu3**	**bleu4**	**rouge1**	**rouge2**	**rougeL**
txt							
r16-a16	0.198	0.108	0.058	0.032	0.298	0.104	0.229
r32-a64	0.193	0.102	0.059	0.034	0.299	0.107	0.231
r128-a128	0.190	0.109	0.072	0.043	0.294	0.105	0.227
RAG	0.155	0.101	0.074	0.055	0.256	0.128	0.201
Base	0.019	0.009	0.005	0.002	0.041	0.012	0.036
pdf							
r16-a16	0.212	0.110	0.072	*0.048*	0.302	0.109	0.235
r32-a64	0.217	0.112	0.068	0.040	0.307	0.109	0.239
r128-a128	***0.218***	*0.117*	*0.073*	0.044	***0.312***	*0.118*	***0.244***
RAG	0.193	**0.124**	**0.092**	**0.068**	0.301	**0.146**	0.238
Base	0.019	0.008	0.004	0.002	0.040	0.011	0.032

Table 5: BLEU & ROUGE mean Scores for both datasets

`pdf` Dataset outperforms `txt` in all metrics. Interestingly larger rank and α seem to boost performance in a lot of these metrics. The standard deviation for ROUGE values were quite high with about 10%. For BLEU the standard deviation was a bit lower with about 8%.

Models	**BERT-Precision**	**BERT-Recall**	**BERT-F1**
txt			
r16-a16	0.869	0.838	0.853
r32-a64	***0.871***	0.838	0.854
r128-a128	0.869	0.837	0.853
RAG	0.792	0.854	0.822
Base	0.751	0.814	0.781
pdf			
r16-a16	0.866	0.844	0.855
r32-a64	0.867	0.845	0.855
r128-a128	0.869	*0.846*	***0.857***
RAG	0.818	**0.858**	0.837
Base	0.746	0.815	0.779

Table 6: BERT Precision, Recall, and F1 scores for both datasets

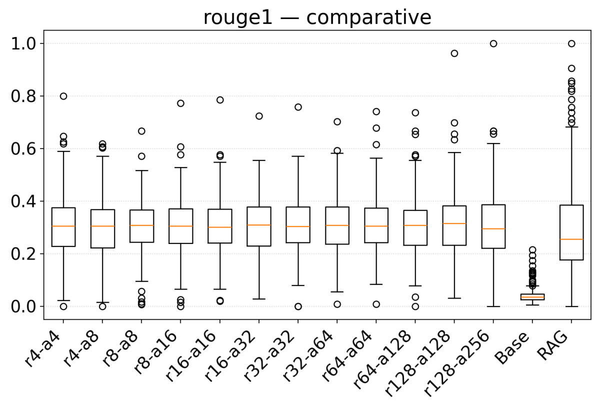
The results show that the `pdf` dataset outperforms the `txt` in two of three metrics while being only slightly worse in the third.

Models	**MMLU Overall**	**SuperGLUE Avg**	**TruthfulQA ACC**
txt			
r16-a16	**0.638**	0.731	0.511
r32-a64	0.628	0.708	0.518
r128-a128	0.615	**0.732**	0.495
RAG	**0.638**	0.716	**0.529**
Base	**0.638**	0.716	**0.529**
pdf			
r16-a16	0.636	0.716	0.498
r32-a64	0.633	0.718	0.490
r128-a128	0.619	0.713	0.478
RAG	**0.638**	0.716	**0.529**
Base	**0.638**	0.716	**0.529**

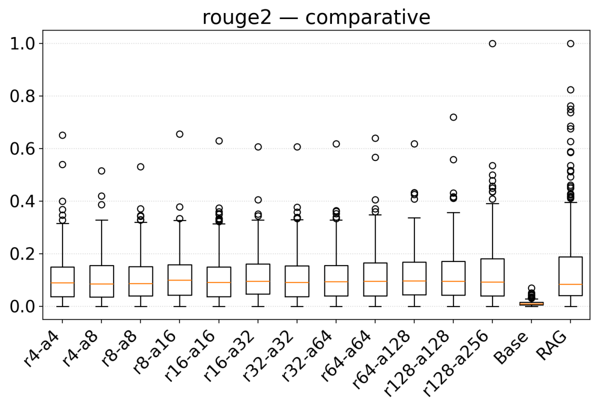
Table 7: General capabilities benchmarks for both datasets

The results for the Base and RAG models are the same since there is no training/fine-tuning for the RAG models. Again these are mean values.

C.2 Supplementary Boxplots



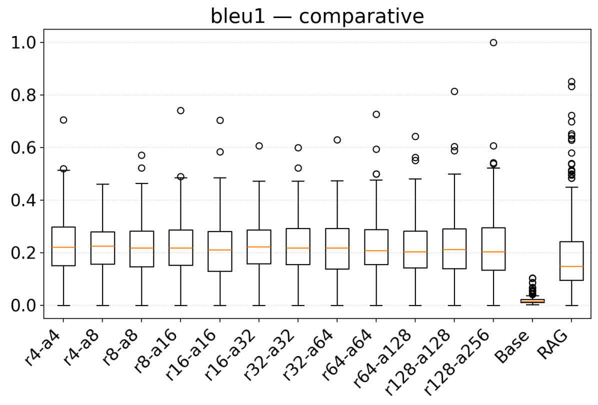
(a) ROUGE 1



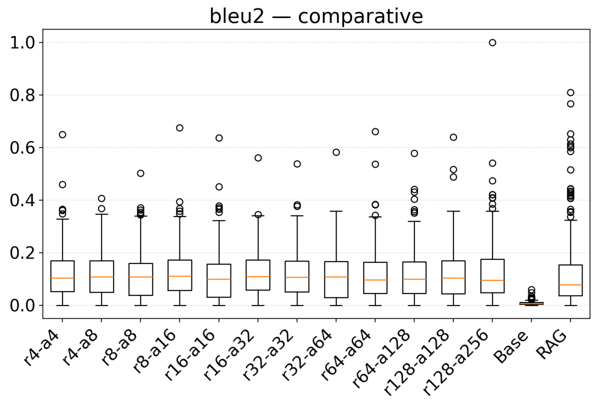
(b) ROUGE 2

Figure 6: ROUGE 1 and ROUGE 2 boxplots

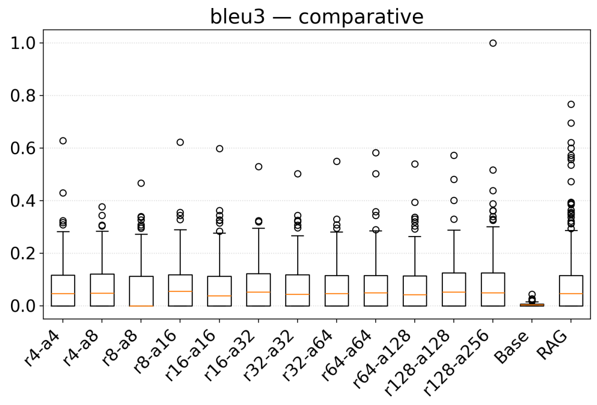
ROUGE 1 and 2 show similar results as ROUGE L as mentioned in Figure 2a. Again, not correlating with the other quality metrics.



(a) BLEU 1



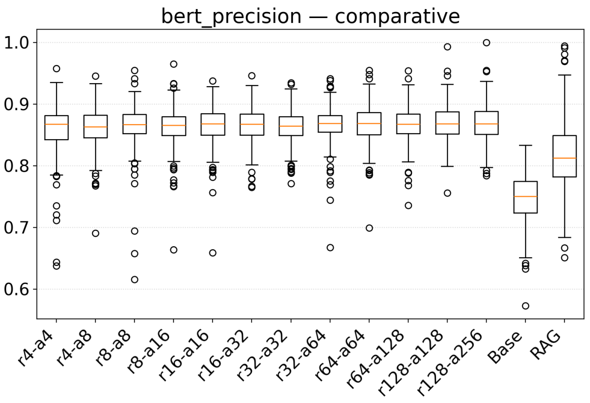
(b) BLEU 2



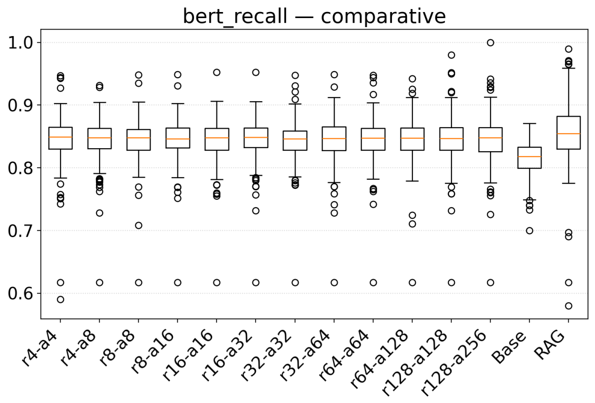
(c) BLEU 3

Figure 7: Boxplots of BLEU 1, 2, and 3 scores

BLEU 1, 2, and 3 show similar results as BLEU 4 as mentioned in Figure 2b. Again, not correlating with the other quality metrics.



(a) BERT Precision



(b) BERT Recall

Figure 8: BERT Precision and Recall boxplots

BERT-precision and recall show similar performance, therefore the BERT-F1 gives a good overall measure of performance, as shown in Figure 2c