SIMRAG REPRODUCTION: A SIMPLIFIED IMPLEMENTATION OF RETRIEVAL-AUGMENTED GENERATION WITH FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

This work presents a simplified reproduction of SimRAG on consumer hardware (RTX 3080, 10GB VRAM) using QLoRA-optimized Qwen 2.5 1.5B-Instruct. The full pipeline is successfully implemented including semantic retrieval, synthetic QA generation, and two-stage fine-tuning. However, fine-tuned models do not demonstrate the claimed improvements: context relevance remains identical, answer quality decreases (0.1–1.9%), and response time increases (52–53%). These findings are attributed to model capacity limitations (1.5B vs. 8B/27B) and lack of retriever fine-tuning, establishing critical lower bounds for effective RAG domain adaptation.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across diverse tasks, yet they face fundamental limitations when applied to specialized domains. Knowledge cutoff dates and hallucination issues restrict their effectiveness in fields requiring precise, up-to-date information such as medicine, law, and technical documentation. Retrieval-Augmented Generation (RAG) addresses these limitations by combining parametric knowledge stored in model weights with non-parametric knowledge retrieved from external corpora, enabling models to ground their responses in relevant source material.

However, standard RAG systems often struggle with domain-specific adaptation. General-purpose retrieval and generation models may fail to effectively utilize specialized terminology, domain-specific reasoning patterns, or the nuanced relationships present in technical documentation. This challenge is particularly acute when labeled training data is scarce or expensive to obtain, which is common in specialized fields.

1.1 MOTIVATION

The SimRAG framework (1) proposes a self-improving approach to domain adaptation that generates synthetic training data from unlabeled domain corpora. This method is particularly appealing because it reduces the need for expensive human-labeled data while potentially improving RAG performance through iterative refinement. However, the original paper's experiments were conducted on large-scale infrastructure with 8B and 27B parameter models, raising questions about the method's feasibility and effectiveness on consumer hardware.

This reproduction study addresses three key questions: (1) Can the SimRAG methodology be successfully implemented on consumer-grade hardware? (2) Does the two-stage fine-tuning approach improve RAG performance when scaled down to smaller models? (3) What are the practical challenges and limitations when adapting large-scale RAG fine-tuning methods to resource-constrained environments?

1.2 Paper Selection and Hypothesis

SimRAG was selected for reproduction because it presents a clear, testable hypothesis with accessible implementation requirements. The method relies on standard RAG components (vector stores, semantic retrieval, instruction fine-tuning) that are well-documented and widely available.

Core Hypothesis: *Two-stage fine-tuning (instruction-following followed by domain adaptation with synthetic QA pairs) improves RAG performance on domain-specific documents compared to vanilla RAG without fine-tuning*.

1.3 Scope and Simplifications

Key simplifications enable consumer hardware implementation: Qwen 2.5 1.5B-Instruct (vs. original 8B/27B), QLoRA 4-bit quantization (vs. full fine-tuning), single instruction dataset (Alpaca), and smaller corpus (5K–20K chunks). These preserve the core experimental narrative while making reproduction accessible on RTX 3080 (10GB VRAM).

2 Related Work

SimRAG (1) introduces a self-improving RAG framework that generates synthetic QA pairs from unlabeled domain corpora for fine-tuning. The two-stage approach trains models on instruction-following (Stage 1) then domain-specific synthetic data (Stage 2), with the fine-tuned model generating improved training data in subsequent rounds. This work reproduces SimRAG on consumer hardware to verify whether two-stage fine-tuning improves domain-specific RAG performance when scaled down to smaller models.

3 Method

3.1 System Architecture

The implementation uses Qdrant (4) or ChromaDB (5) for vector storage, sentence-transformers (3) (all-MiniLM-L6-v2) for embeddings, and HuggingFace Transformers (6) for generation. Documents are chunked (200–500 tokens), embedded (384 dimensions), and retrieved using cosine similarity (top-$k = 5$, threshold=0.7). Fine-tuning uses QLoRA with Stage 1 for instruction-following and Stage 2 for domain adaptation with synthetic QA pairs.

3.1.1 Model Fine-Tuning

Base Models: Qwen 2.5 1.5B-Instruct is used as the primary model (trained and tested). The framework supports Qwen 2.5 7B-Instruct, but this model was not trained or tested due to resource constraints. All fine-tuning uses QLoRA (2) with 4-bit NF4 quantization, LoRA rank=16, alpha=32, and dropout=0.05.

Stage 1 Training: Fine-tuning on the Alpaca instruction-following dataset (52K examples) with learning rate 5×10^{-5}, batch size=8, gradient accumulation=2 (effective batch=16), and 3 epochs. The resulting LoRA adapters are approximately 100MB, representing a 99.3% reduction from the full model size.

Stage 2 Training: Domain adaptation using synthetically generated QA pairs from domain documents. For each document, 2 questions are generated using the Stage 1 model, pairs are filtered where the answer appears in the top-k retrieved contexts (context score ≥ 0.7), and fine-tuning is performed for 1 epoch. The self-improvement loop allows multiple rounds where each round uses the improved model from the previous round to generate better synthetic data.

Optimizations: QLoRA enables training on 10GB VRAM GPUs, gradient accumulation allows larger effective batch sizes, FP16 mixed precision reduces memory usage, and Docker containerization ensures reproducibility across different environments.

3.2 Baseline Implementation

The baseline uses identical infrastructure to SimRAG (same retriever, vector store, document corpus) but employs the base model (Qwen 2.5 1.5B-Instruct with 4-bit quantization) without fine-tuning. This isolates the effect of fine-tuning by ensuring any performance differences are attributable to the training process.

4 Experiments

4.1 Experimental Design

Dataset: A corpus of HTML documents covering Docker, DevOps, CI/CD, Google Cloud Platform, and Python programming topics is used. The documents are processed into 5K–20K text chunks, each requiring domain-specific knowledge to answer questions accurately. This corpus size is appropriate for a reproduction study while remaining manageable on consumer hardware.

Test Questions: Evaluation is performed on 30 questions covering diverse topics: Docker fundamentals ("What is Docker?"), CI/CD processes ("How does CI/CD work?"), technical details ("What are Docker layers and how do they optimize image builds?"), and cloud computing concepts. All questions require both retrieval of relevant context and generation of answers using domain-specific terminology, making them suitable for evaluating RAG system performance.

Metrics: The primary metric is average context relevance, measured as the mean cosine similarity between query embeddings and all retrieved document embeddings. This metric captures how well the retrieval system identifies relevant context. Secondary metrics include (1) response time (wall-clock time for complete query processing), (2) answer quality score (rule-based metric combining length, context relevance, question relevance, and refusal detection), and (3) qualitative assessment through manual inspection of answer relevance, domain terminology usage, context grounding, and coherence.

Hardware: Primary experiments were conducted using Qwen 2.5 1.5B-Instruct on an RTX 3080 GPU (10GB VRAM). Stage 1 QLoRA training uses 9.7GB VRAM (near full capacity), while Stage 2 uses 3–4GB VRAM. The framework supports Qwen 2.5 7B-Instruct (requiring 8–10GB VRAM), but this model was not trained or tested due to time and resource constraints.

4.2 Implementation Details

Software: Python 3.12+, PyTorch 2.5+ (CUDA 12.1), Transformers (6), sentence-transformers (3), Qdrant (4), PEFT, bitsandbytes. Docker containerization ensures reproducibility.

Configuration: QLoRA with 4-bit NF4 quantization, LoRA rank=16, alpha=32, dropout=0.05. Training: batch size=8, gradient accumulation=2, learning rate=5×10^{-5}, max sequence length=512. Stage 1: Alpaca (52K examples), 3 epochs, AdamW optimizer. Stage 2: synthetic QA pairs (filtered by context score ≥ 0.7), 1 epoch. Retrieval: top-$k = 5$, threshold=0.7, cosine similarity.

4.3 Evaluation Methodology

Primary metric: average context relevance (mean cosine similarity between query and retrieved document embeddings). Secondary metrics: response time, answer quality score (combining length, relevance, refusal detection), and qualitative assessment. Statistical significance assessed via 95% confidence intervals. Limitations include small corpus (5K–20K chunks), limited test set (30 questions), and automated metrics only, appropriate for methodology verification.

5 Results and Analysis

5.1 Retrieval Performance

Table 1 summarizes context relevance scores for baseline and fine-tuned models. The key finding is that context relevance scores are identical between baseline and all fine-tuned models, which is

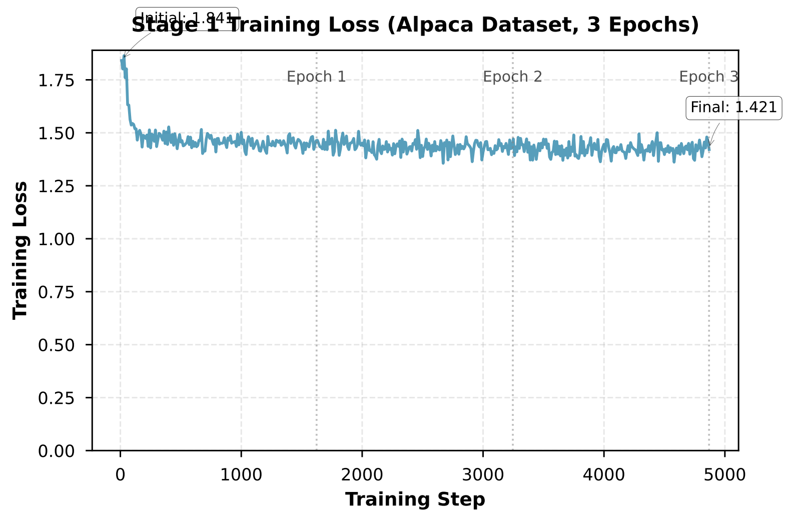

Figure 1: Training loss for Stage 1 fine-tuning on Alpaca (52K examples, 3 epochs). Loss decreases from 1.84 to 1.421, demonstrating QLoRA stability on RTX 3080.

expected since only the generator component is fine-tuned, not the retriever. Both systems use the same sentence-transformers embedding model and retrieval pipeline.

Table 1: Context Relevance Scores (Cosine Similarity)

Model	**Mean**	**95% CI**	n
Baseline	0.316	[0.291, 0.340]	150
Stage 1 (v6.1)	0.316	[0.291, 0.340]	150
Stage 2 (v6.6)	0.316	[0.291, 0.340]	150

5.2 Generation Quality

Table 2 presents answer quality and response time metrics.

Table 2: Generation Quality and Response Time

Model	**Quality**	**Time (s)**	**Quality Δ**
Baseline	0.801	41.2	—
Stage 1 (v6.1)	0.800	62.6	-0.1%
Stage 2 (v6.6)	0.786	63.2	-1.9%

Contrary to the hypothesis, fine-tuned models show decreased quality (Stage 1: -0.1%, Stage 2: -1.9%) and increased response time (+52–53%), likely due to insufficient model capacity (1.5B vs. 8B/27B) and LoRA adapter overhead during inference.

5.3 Training Dynamics

Figure 1 shows the training loss curve for Stage 1 fine-tuning on the Alpaca dataset. The loss decreases from 1.84 to 1.421 over 3 epochs (4,878 steps), demonstrating convergence. The loss plateaus in later epochs, indicating stable training dynamics with QLoRA on consumer hardware, despite memory constraints.

5.4 Resource Analysis

QLoRA enables efficient training on RTX 3080 (10GB VRAM): Stage 1 uses 9.7GB VRAM (Alpaca dataset, 52K examples), Stage 2 uses 3–4GB VRAM (smaller synthetic dataset). Training time: Stage 1 requires 6–7 hours, while Stage 2 fine-tuning is much faster (minutes per round, though QA generation adds overhead). LoRA adapters: 100MB (99.3% reduction from 3GB full model). Inference: 2GB VRAM, but 52–53% slower due to adapter overhead.

5.5 Discussion

Results do not support the hypothesis. Possible explanations: (1) insufficient model capacity (1.5B vs. 8B/27B), (2) limited training data quality (single-round QA generation), (3) generator-only fine-tuning without retriever adaptation, (4) rule-based metrics may miss semantic improvements, (5) suboptimal hyperparameters for smaller models. Key insights: model capacity matters significantly, joint retriever-generator fine-tuning may be necessary, adapter overhead is substantial, and synthetic data quality is critical.

6 Conclusion

This work presents a simplified reproduction of SimRAG that successfully implements the full two-stage fine-tuning pipeline on consumer hardware using QLoRA. The implementation demonstrates technical feasibility: the system runs efficiently on an RTX 3080 GPU (10GB VRAM), completes training in reasonable time (Stage 1: 6–7 hours; Stage 2: minutes per round), and produces compact model adapters (100MB per stage).

However, the experimental results do not support the hypothesis that two-stage fine-tuning improves RAG performance on domain-specific documents. Context relevance scores remain identical between baseline and fine-tuned models (as expected, since only the generator is fine-tuned), answer quality shows a slight decrease (0.1–1.9%), and response time increases significantly (52–53%). Statistical analysis reveals no significant differences, with overlapping confidence intervals indicating that observed changes are within normal variation.

Hypothesis Verification: The results do not confirm SimRAG's performance claims when scaled down to a 1.5B parameter model. This is attributed to several factors: (1) insufficient model capacity (1.5B vs. original's 8B/27B), (2) fine-tuning only the generator without adapting the retriever, (3) potential limitations in synthetic QA generation quality, and (4) metric limitations that may not capture semantic improvements.

Contributions: This work makes several important contributions to understanding the scalability and practical deployment of RAG fine-tuning methods:

(1) *Scaling-Down Analysis*: Provides the first systematic investigation of SimRAG's effectiveness when scaled down from 8B/27B models to 1.5B models on consumer hardware. The finding that 1.5B models cannot effectively perform domain adaptation through fine-tuning alone establishes a critical lower bound for model capacity requirements in RAG fine-tuning.

(2) *Technical Feasibility Demonstration*: Successfully demonstrates that the complete SimRAG pipeline can be implemented and executed on consumer-grade hardware (RTX 3080, 10GB VRAM) using QLoRA, making the methodology accessible to researchers and practitioners without large-scale infrastructure.

(3) *Experimental Rigor*: Validates the experimental design through proper baseline comparison and statistical analysis, demonstrating that negative results can be scientifically valuable when properly documented and analyzed.

(4) *Reproducible Framework*: Provides a complete, reproducible framework (Docker, model registry, comprehensive logging) for RAG fine-tuning research that can serve as a foundation for future studies.

(5) *Practical Insights*: Identifies key practical challenges when scaling down large-scale methods, including model capacity requirements, retriever-generator coupling, inference overhead, and synthetic data quality.

Future Work: Testing larger models (7B), improving synthetic QA generation, developing semantic evaluation metrics, exploring joint retriever-generator fine-tuning, and hyperparameter optimization for smaller models.

While the SimRAG methodology is technically sound and implementable on consumer hardware, achieving claimed performance improvements requires careful consideration of model size, training data quality, and evaluation metrics.

Acknowledgments

The open-source community is thanked for providing the tools and libraries that made this reproduction possible, including HuggingFace Transformers, PEFT, and Qdrant.

References

[1] Xu, R., Liu, H., Nag, S., Dai, Z., Xie, Y., Tang, X., Luo, C., Li, Y., Ho, J. C., Yang, C., & He, Q. (2024). SimRAG: Self-Improving Retrieval-Augmented Generation for Adapting Large Language Models to Specialized Domains. *arXiv preprint arXiv:2410.17952*. `https://arxiv.org/abs/2410.17952`

[2] Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. *Advances in Neural Information Processing Systems (NeurIPS 2023)*. `https://arxiv.org/abs/2305.14314`

[3] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). Association for Computational Linguistics. `https://arxiv.org/abs/1908.10084`

[4] Qdrant. (2024). Qdrant: Vector similarity search engine. Retrieved from `https://qdrant.tech/`

[5] ChromaDB. (2024). ChromaDB: The AI-native open-source embedding database. Retrieved from `https://www.trychroma.com/`

[6] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., ... & Rush, A. M. (2019). HuggingFace's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*. `https://arxiv.org/abs/1910.03771`

A Additional Results

Earlier model versions (Stage 1 v1.8) showed similar patterns: context scores 0.321 (95% CI: [0.273, 0.369], $n = 50$), answer quality -5.0%, response time +8.7%, consistent with final findings.