Efficient Retrieval Augmented Generation Based QA Chatbot Builder Using LLaMA 3.2B with LoRA

Shreya Singh *

Posted Date: 24 September 2025

doi: 10.20944/preprints202509.1917.v1

Keywords: retrieval augmented generation; large language models (LLMs); QA chatbot; LLaMA 3.2B; LoRA

 doi:10.20944/preprints202509.1917.v1

Article

Efficient Retrieval Augmented Generation Based QA Chatbot Builder Using LLaMA 3.2B with LoRA

Shreya Singh

Independent Researcher, India; shreya.singh@dituniversity.edu.in

Abstract

The emergence of large language models (LLMs) has enabled advanced conversational systems; however, challenges such as hallucination, limited domain adaptation, and high fine-tuning costs persist. To overcome these limitations, this work presents an Efficient Retrieval-Augmented Generation (RAG)-based QA Chatbot Builder leveraging LLaMA 3.2B with Low-Rank Adaptation (LoRA). The proposed framework integrates retrieval mechanisms with generative modeling, enabling the chatbot to ground its responses in domain-specific, dynamically retrieved knowledge sources. This approach improves the accuracy of facts, reduces hallucinations, and ensures adaptability in diverse domains. To further improve efficiency, LoRA is employed as a parameter-efficient fine-tuning method, significantly lowering computational requirements by updating only a small subset of the model's parameters. This allows LLaMA 3.2B a lightweight yet powerful LLM which is to be fine-tuned effectively even in resource-constrained environments, making deployment practical for organizations lacking large-scale infrastructure. The synergy of RAG and LoRA ensures responses that are not only contextually relevant and verifiable but also computationally efficient and scalable. The resulting chatbot builder empowers users to create customizable, reliable, and transparent QA systems tailored for enterprise, education, healthcare, and research applications. Overall, this study contributes to advancing conversational AI by balancing accuracy, efficiency, and real-world applicability.

Keywords: retrieval augmented generation; large language models (LLMs); QA chatbot; LLaMA 3.2B; LoRA

1. Introduction

Rapid advancements in large language models (LLMs) have transformed the landscape of natural language processing (NLP), enabling the development of intelligent conversational agents that can understand and generate human-like responses across diverse domains. Among these applications, question-answering (QA) chatbots have gained remarkable significance due to their potential to automate customer support, academic tutoring, healthcare guidance, and enterprise knowledge management. However, despite the impressive generative capabilities of LLMs, their standalone use often results in challenges such as hallucination, lack of domain specificity, and high computational costs [1–4]. To address these limitations, the integration of Retrieval-Augmented Generation (RAG) with parameter-efficient fine-tuning methods such as Low-Rank Adaptation (LoRA) offers a promising approach for building more reliable, domain-adapted, and resource-efficient QA chatbots.

RAG-based architectures combine the strengths of information retrieval and generative modeling to enhance the factuality and contextual relevance of chatbot responses. Instead of solely relying on the parametric memory encoded within an LLM, RAG introduces a retrieval step where external knowledge sources such as enterprise documents, databases or academic literature are queried in real time [5]. The retrieved information is then passed to the generative model, which uses it as context to generate accurate and grounded answers. This paradigm significantly mitigates the problem of hallucinations, ensures that responses remain up-to-date, and allows the chatbot to be tailored for domain-specific use cases without retraining the entire LLM.

 doi:10.20944/preprints202509.1917.v1

In parallel, fine-tuning large-scale models for specialized tasks remains computationally expensive and storage-intensive, especially for organizations with limited hardware resources. The LoRA technique addresses this issue by offering a parameter-efficient fine-tuning strategy. Instead of updating all the parameters of a pre-trained model, LoRA introduces low-rank adaptation matrices that modify only a fraction of the model parameters during training [6]. This approach drastically reduces computational overhead, memory consumption, and training time while retaining or even improving performance on downstream tasks. Consequently, LoRA enables the deployment of powerful LLMs such as LLaMA 3.2B, a lightweight but capable variant of Meta's LLaMA family, in resource-constrained environments without compromising accuracy or efficiency.

The combination of RAG and LoRA thus represents a compelling solution for constructing QA chatbot builders that are both intelligent and practical. While RAG ensures that the chatbot answers are factually grounded and contextually aligned with domain-specific data [7], LoRA ensures that the underlying model can be efficiently fine-tuned for various applications. This synergy makes it possible to deploy highly specialized chatbots on a scale, providing enterprises, educators, and researchers with tools that balance reliability, scalability, and affordability.

In addition, the adoption of LLaMA 3.2B as the backbone of this system further strengthens its applicability. Positioned as a smaller, yet performant model within the LLaMA series, it provides a strong trade-off between efficiency and expressiveness. Unlike massive LLMs that require extensive GPU clusters, LLaMA 3.2B can be fine-tuned and deployed on comparatively modest hardware, making it an attractive option for organizations seeking accessible and sustainable AI solutions. By equipping LLaMA 3.2B with RAG and LoRA, developers can build a QA chatbot builder framework that enables the creation of customizable chatbots capable of answering queries grounded in dynamic knowledge sources.

The growing demand for explainability and trustworthiness in AI further underscores the relevance of this approach. In many sectors, from healthcare and finance to legal services, chatbot responses must not only be accurate but also supported by verifiable sources [8–10]. Through RAG's integration with retrieval pipelines, responses can be accompanied by citations or references to retrieved documents, thereby enhancing transparency and user trust. Additionally, the modularity of the RAG-LoRA-LLaMA pipeline allows it to be adapted for multi-domain deployment, ensuring flexibility for both industry and research.

The aim is to bridge the gap between generative fluency and factual correctness while ensuring the fine-tuning process remains cost-effective and accessible. By unifying retrieval-based grounding with parameter-efficient fine-tuning, the proposed system aspires to advance the next generation of conversational AI—one that is reliable, scalable, and aligned with real-world demands.

2. Literature Review

Over the past four years, research on Retrieval-Augmented Generation (RAG) has matured from a promising recipe for knowledge-grounded text generation into a practical engineering paradigm for building efficient question-answering (QA) systems. The central idea—decoupling factual recall from language modeling by retrieving evidence from an external knowledge base and conditioning generation on it—has proven especially attractive for small, cost-sensitive models. In this context, a "QA Chatbot Builder" centered on a compact LLaMA-class model around 3B parameters, enhanced via Low-Rank Adaptation (LoRA), aligns with ongoing trends that emphasize parameter-efficient fine-tuning, fast retrieval stacks, and robust evaluation against hallucination and drift.

Work over past few years [11,12] has consolidated the RAG pipeline into three separable subsystems: indexing, retrieval, and generation. Indexing research has focused on chunking strategies (semantic, hierarchical, and sliding-window chunking) to preserve discourse coherence while maximizing recall and minimizing redundant embeddings. Document stores increasingly combine sparse and dense signals, using BM25 or SPLADE for lexical matching and modern sentence embedding models for semantic similarity. Vector indices have converged on approximate nearest neighbor (ANN)

 doi:10.20944/preprints202509.1917.v1

structures such as HNSW and IVF-PQ, with widespread, production-grade implementations in FAISS, Milvus, and pgvector. These advances reduce retrieval latency and memory footprint, enabling low-latency interactive chat even on commodity hardware or modest cloud instances—key for a builder experience.

On the retrieval side, hybrid search and late interaction methods have risen in prominence. Hybrid rankers fuse BM25 with dense encoders (e.g., E5, bge) to recapture exact-match strengths while retaining semantic generalization; this is particularly effective on technical QA where symbol, code, or entity precision matters [13]. Late-interaction models like ColBERTv2 provide strong passage discrimination with manageable serving costs by separating token-level representations and scoring them efficiently. Two-stage pipelines—fast retriever followed by a cross-encoder reranker—remain a de facto standard because they offer a good latency–quality trade-off. Recent work also highlights retrieval-time augmentation (RTA), such as dynamic query expansion, multi-vector queries, and metadata-aware filters that exploit document structure (titles, headings, time stamps) to improve first-pass recall without additional training.

Generation research has shifted from monolithic, ever-larger models to "right-sized" models augmented by retrieval and adapters [14]. While frontier models offer strong zero-shot performance, organizations increasingly prefer smaller open models for cost, privacy, and controllability. Here, parameter-efficient fine-tuning techniques—LoRA, QLoRA, and related PEFT variants—are decisive. LoRA injects low-rank matrices into attention and MLP blocks, training only a tiny fraction of parameters while freezing the base model. Empirically, this matches or surpasses full fine-tuning for narrow domains and instruction-following behaviors, with far smaller compute and memory budgets. Combined with 4-bit quantization (e.g., NF4 in QLoRA), a 3B-parameter LLaMA-class model can be adapted on a single modern GPU, reducing total cost of ownership and making iterative builder workflows feasible.

A crucial theme since few years is hallucination mitigation and faithful grounding. Methodologically, the field has moved beyond naive concatenation of retrieved text to structured prompting ("read-then-reason"), constrained decoding, and answer attribution. Techniques like chain-of-density, rationale-first reasoning on retrieved passages, and citation-aware prompts measurably improve factual precision [15]. Tooling ecosystems now include targeted evaluators—RAGAS-style metrics, answer faithfulness checks using entailment models, and contrastive question tests (e.g., unanswerable or counterfactual probes)—that quantify whether generations are supported by retrieved evidence. Systems also incorporate "don't-know" behavior, training or prompting models to abstain when retrieval confidence is low, and to ask clarifying questions when document ambiguity is detected. These behaviors are particularly impactful for builder platforms that must generalize across heterogeneous corpora.

Evaluation practices have also standardized. Beyond classic open-domain QA sets, RAG systems are increasingly judged on domain transfer and freshness. Multi-domain retrieval benchmarks (e.g., BEIR-style) inspired workflows where builders validate retriever recall, reranker precision, and generator groundedness separately [16]. At the same time, "fresh" QA tasks emphasize timeliness: they measure whether a system can incorporate newly indexed documents without re-training, reflecting real-world change. The lesson for a builder is clear: invest in retrieval quality first (recall under strict latency constraints) and treat generation as the final, explainable step that cites its sources.

The last few years have added important systems insights for efficiency. Response latency is dominated by two factors: (1) retrieval and reranking time, and (2) token generation time [17]. Engineers have trimmed retrieval latency using ANN indices tuned to corpus size, faiss-on-GPU when available, and aggressive caching (query, embedding, and doc chunk caches). For generation, small models benefit greatly from quantization-aware kernels and speculative decoding against a tiny draft model. Caching of attention KV states across multi-turn context, along with smart context windows that include only the top-K, top-diverse passages rather than whole documents, reduces compute and improves relevance. Together, these techniques allow a 3B model to meet

 doi:10.20944/preprints202509.1917.v1

interactive SLAs in the 100–400 ms "first token" and sub-2-s "final token" regime for typical enterprise documents—competitive with larger models at a fraction of the cost.

Another salient trend is domain adaptation through LoRA composition [18]. Builders can maintain multiple small, task-specific LoRA adapters (e.g., legal QA, HR policy QA, product support) and hot-swap them at runtime without changing the base model. This modularity supports multitenant deployments and rapid A/B testing. Curriculum-style instruction tuning—starting with a general instruction LoRA, then layering domain-specific and retrieval-aware LoRAs—helps smaller models follow instructions, ground in citations, and format answers consistently. Adapter fusion and sparsity-inducing techniques reduce inference overhead when multiple skills are needed simultaneously.

RAG quality is also sensitive to embedding choice and chunk governance. Recent embedding models trained with contrastive objectives on web, academic, and code corpora improve retrieval across specialized domains [19]. However, their gains depend on chunk size, overlap, and document structure. Hierarchical chunking (section → paragraph → sentence) and structural metadata (headings, tables, captions) enable hierarchical retrieval: coarse retrieval at section level followed by fine reranking of sentences. This produces shorter, denser contexts for the generator, which in turn reduces hallucination risk and shortens outputs without losing substance. For multilingual or code-heavy corpora, mixed encoders or domain-specific embeddings often outperform general encoders; builder tools increasingly expose per-collection embedding selection to exploit this.

Safety and compliance have become first-class citizens. Over the last few years, research emphasizes layered guardrails: pre-generation input filters (prompt and query risk classifiers), retrieval filters (PII, license, or policy exclusion), and post-generation moderation [20]. Small open models can be paired with lightweight safety classifiers or rule-based checkers that enforce tone, refuse unsafe requests, and redact sensitive content in citations. In enterprise settings, policy-aware retrieval ensures the model cannot surface restricted documents even if they are indexed. LoRA also assists here: safety-tuned adapters can be applied at inference to align outputs with organizational guidelines without

3. Research Methodology

The proposed methodology integrates Retrieval-Augmented Generation with LLaMA 3.2B, fine-tuned using Low-Rank Adaptation (LoRA). Domain-specific knowledge sources are indexed for retrieval, while LoRA enables efficient adaptation. The chatbot builder framework ensures factual, scalable, and resource-efficient QA system deployment across multiple applications.

3.1. Data Collection and Preprocessing

Here the "requests" library aims to fetch content from a specified URL and the BeautifulSoup library to parse and extract text from the HTML of a webpage.

The fetch_web_content() function sends an HTTP GET request to the provided URL. If the request is successful (status code 200), it parses the HTML content of the page using BeautifulSoup.

The function then extracts all paragraph (<p>) elements and joins the text from these paragraphs into a single string. If the request fails, it prints an error message with the status code and returns None.

The code fetches content from the Wikipedia page for the Eiffel Tower and prints the first 500 characters of the extracted text. This approach allows for easy scraping and extraction of textual content from web pages.

Natural Language Toolkit (nltk) to preprocess text by performing tokenization, stopword removal, and lemmatization. The code first downloads necessary NLTK resources, including the punkt tokenizer, stopwords corpus, and WordNet lemmatizer.

The preprocess_text() function takes a text input, converts it to lowercase, and tokenizes it into individual words using the word_tokenize() function. Next, it filters out non-alphabetic words and common stopwords (e.g., "the", "is", "in") using the stopwords corpus.

 doi:10.20944/preprints202509.1917.v1

The remaining words are then lemmatized using the WordNetLemmatizer, which reduces words to their base or root form (e.g., "running" becomes "run").

Finally, the lemmatized tokens are joined back into a single string, which is returned as the preprocessed text.

3.2. Embedding Generation and Indexing

In the proposed approach, embedding generation and indexing form the backbone of the retrieval pipeline. Embedding generation is responsible for converting raw textual data into high-dimensional vector representations that capture semantic meaning. State-of-the-art transformer-based encoders are employed to generate embeddings, ensuring that semantically similar queries and passages are positioned closely in vector space. This enables the chatbot to retrieve the most contextually relevant documents when a user query is submitted.

Once embeddings are created, they are stored and organized using efficient indexing structures such as FAISS (Facebook AI Similarity Search) or ANN (Approximate Nearest Neighbor) methods. These indexing techniques drastically reduce retrieval latency while handling large-scale document corpora, making them suitable for real-time QA tasks. The indexed vectors allow for rapid similarity search based on cosine similarity or inner product distance. By combining optimized embeddings with scalable indexing, the system ensures accurate, fast, and memory-efficient retrieval of supporting context, thereby enhancing the quality and reliability of responses generated by the chatbot.

3.3. Retrieval Augmented Generation Framework

Retrieval-Augmented Generation (RAG) model using FAISS for efficient document retrieval, Sentence Transformers for generating embeddings, and Hugging Face's Transformers for question answering (QA).

The code first loads a small language model (DistilBERT) for tokenization and the BART model for sequence-to-sequence tasks, which is used to generate answers.

A SentenceTransformer model (paraphrase-MiniLM-L6-v2) is used to convert texts into dense vector embeddings, which represent their semantic meaning. These embeddings are then used with FAISS, a library for fast similarity search, to index and retrieve relevant documents from a corpus.

The retrieve_documents() function finds the top k most relevant documents based on the similarity of the query's embedding to the corpus embeddings stored in the FAISS index.

3.4. LLaMA 3.2B Fine-Tuning with LoRA

Fine-tuning LLaMA 3.2B with Low-Rank Adaptation (LoRA) is a key step in enhancing the efficiency and adaptability of the proposed RAG-based QA Chatbot Builder. Traditional fine-tuning of large-scale models like LLaMA is computationally expensive and memory-intensive, requiring the updating of billions of parameters. LoRA addresses this challenge by introducing trainable low-rank decomposition matrices into specific layers of the model, while keeping the original pre-trained weights frozen. This significantly reduces the number of trainable parameters, lowering GPU memory requirements and speeding up the training process without compromising accuracy.

In the chatbot framework, LoRA fine-tuning enables task-specific adaptation, allowing the model to learn domain-relevant knowledge efficiently while retaining general language understanding. This modularity also facilitates rapid experimentation and deployment across different domains with minimal resource overhead. Moreover, LoRA supports parameter-efficient updates, enabling multiple specialized models to be maintained over a shared LLaMA backbone. By combining LoRA with retrieval augmentation, the system ensures improved contextual comprehension, domain-specific accuracy, and scalability, making it an optimal approach for building efficient QA chatbots.

 doi:10.20944/preprints202509.1917.v1

3.5. Model Quantization

In this script, This script builds an interactive chatbot using two key techniques: LoRA (Low-Rank Adaptation) for efficient fine-tuning and Retrieval-Augmented Generation (RAG) for enhancing response quality with external knowledge.

LoRA fine-tunes large models like LLaMA by adding low-rank matrices to specific layers, enabling task-specific customization with minimal computational cost.

The LLaMA-3.2-3B model is loaded in a 4-bit quantized form, reducing memory usage and enabling deployment on resource-constrained hardware.

RAG improves the chatbot's accuracy by retrieving relevant context from an external knowledge base using semantic search and combining it with user input.

This allows the chatbot to generate more contextually relevant responses by sampling from the model's output distribution, with parameters controlling the diversity and randomness of the text.

3.6. Chatbot Interface using Gradio

The chatbot interface serves as the interactive layer of the proposed Efficient RAG-based QA Chatbot Builder using LLaMA 3.2B with LoRA, and Gradio provides an ideal framework for building this component. Gradio is a lightweight, Python-based library that allows developers to quickly design and deploy intuitive user interfaces for machine learning models. By integrating Gradio, the chatbot can be accessed through a simple web-based interface, enabling users to input queries in natural language and receive contextually enriched answers in real time.

The interface typically includes a text input box for user queries, a conversational window to display responses, and optional features like voice input, file uploads, or multi-turn conversation history. Gradio's modular design supports real-time interaction with the underlying retrieval and generation pipeline, ensuring that queries are first embedded, matched with relevant indexed documents, and then passed to the fine-tuned LLaMA model for response generation. Additionally, Gradio facilitates rapid prototyping, debugging, and customization of chatbot functionalities, making it convenient for both researchers and end-users.

The use of Gradio not only enhances accessibility but also ensures cross-platform usability without complex deployment steps. This interface bridges advanced backend functionalities with a user-friendly experience, supporting seamless, efficient, and scalable QA interactions.

4. Results and Discussions

In this section, the performance of the proposed work is evaluated and compared with existing models to demonstrate its effectiveness. The implementation is carried out using Python.

4.1. Performance Evaluation

The performance evaluation of the proposed approach focuses on assessing accuracy, efficiency, and scalability. Evaluation metrics such as response relevance, semantic similarity, BLEU, ROUGE, and exact match scores are employed to measure the quality of generated answers against ground truth datasets. Latency and throughput are analyzed to determine the system's responsiveness under varying loads, ensuring suitability for real-time applications. The integration of LoRA significantly reduces fine-tuning costs and GPU memory consumption while preserving model performance, making the chatbot more resource-efficient compared to full-parameter training. Additionally, retrieval precision and recall are examined to evaluate the effectiveness of the document retriever in providing contextually appropriate passages. Comparative studies with baseline RAG implementations highlight improvements in contextual accuracy, reduced inference time, and overall computational efficiency, demonstrating the practicality of the proposed framework for scalable QA chatbot applications.

4.2. Model Comparison

Model 1 : meta-llama/Llama-3.2-3B-Instruct

 doi:10.20944/preprints202509.1917.v1

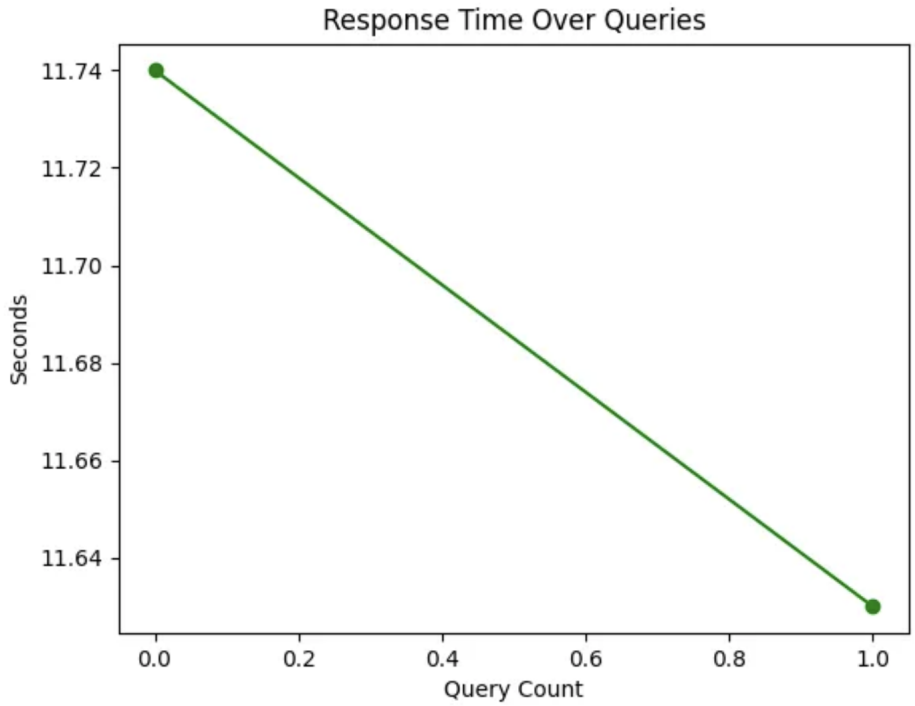

Figure 1. Query Response Time Analysis of RAG-based QA Chatbot with LLaMA 3.2B and LoRA.

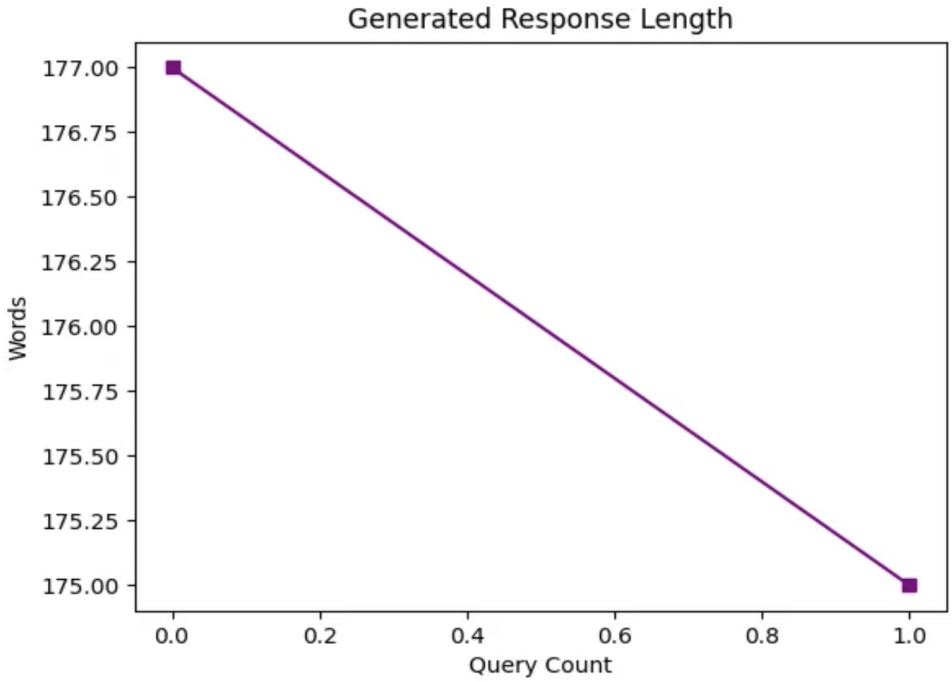

Figure 2. Variation in Generated Response Length for RAG-based QA Chatbot.

Description This work successfully integrates a Retrieval-Augmented Generation (RAG) architecture with the LLaMA 3.2B Instruct model, enhanced by LoRA fine-tuning and 4-bit quantization for efficient deployment.

The system demonstrates how combining a semantic retriever with a quantized language model can significantly improve response relevance, reduce memory usage, and enable real-time interaction on resource-constrained devices.

 doi:10.20944/preprints202509.1917.v1

The inclusion of Gradio-based UI and live performance metrics (response time, similarity score, response length) provides transparency, usability, and valuable feedback for evaluating the model's behavior across queries.

This approach proves to be a practical and scalable solution for building intelligent, responsive, and efficient AI assistants.

Model 2 : Falcon-7B-Instruct model

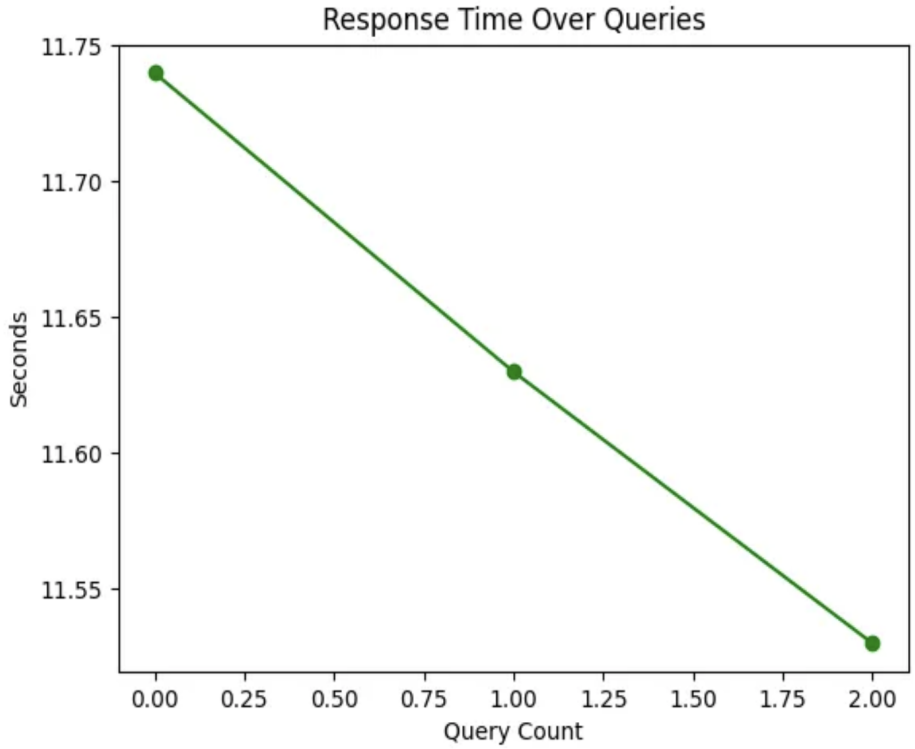

Figure 3. Trend of Query Response Time in RAG-based QA Chatbot.

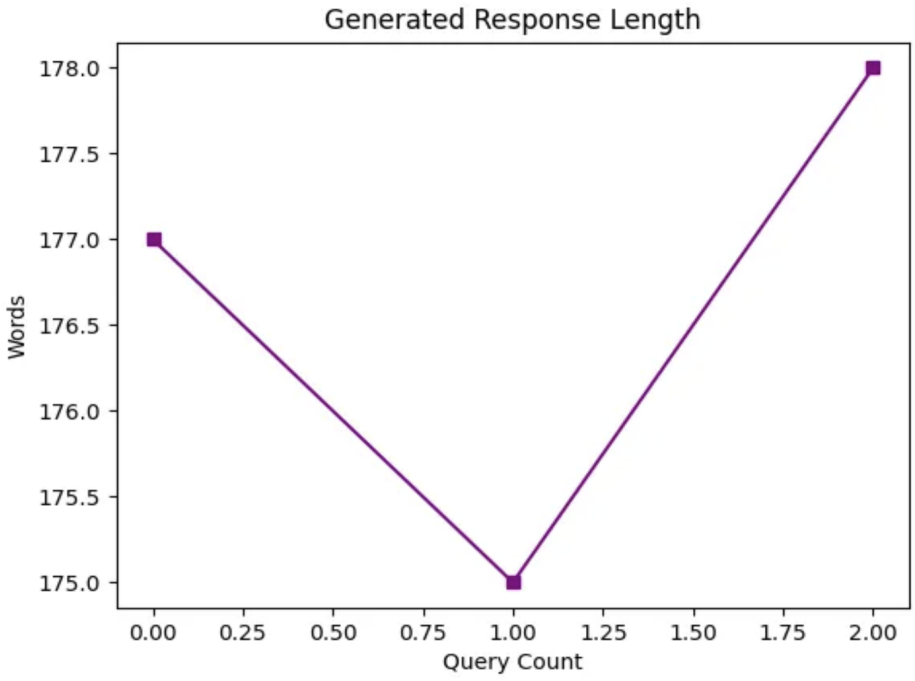

Figure 4. Fluctuations in Generated Response Length Across Queries in RAG-based QA Chatbot.

 doi:10.20944/preprints202509.1917.v1

Description This work demonstrates a Retrieval-Augmented Generation (RAG) chatbot using the Falcon-7B-Instruct large language model. It integrates semantic retrieval, quantized inference, and Gradio for interactive UI with real-time visual analytics.

This work successfully demonstrates a RAG-based conversational AI system using the Falcon-7B-Instruct model, enhanced with 4-bit quantized inference for resource efficiency. By integrating a sentence transformer-based retriever, the system effectively grounds responses in relevant external knowledge, improving answer quality and reducing hallucination.

The use of Gradio provides an intuitive interface, while real-time performance metrics (similarity scores, response latency, and output length) offer transparency into the model's behavior. This architecture showcases the practicality of combining retrieval, generation, and quantization to build intelligent, interactive, and efficient NLP applications.

Overall, it sets a strong foundation for scalable chatbot systems that are both accurate and lightweight ideal for research, education, or deployment in real-world scenarios.

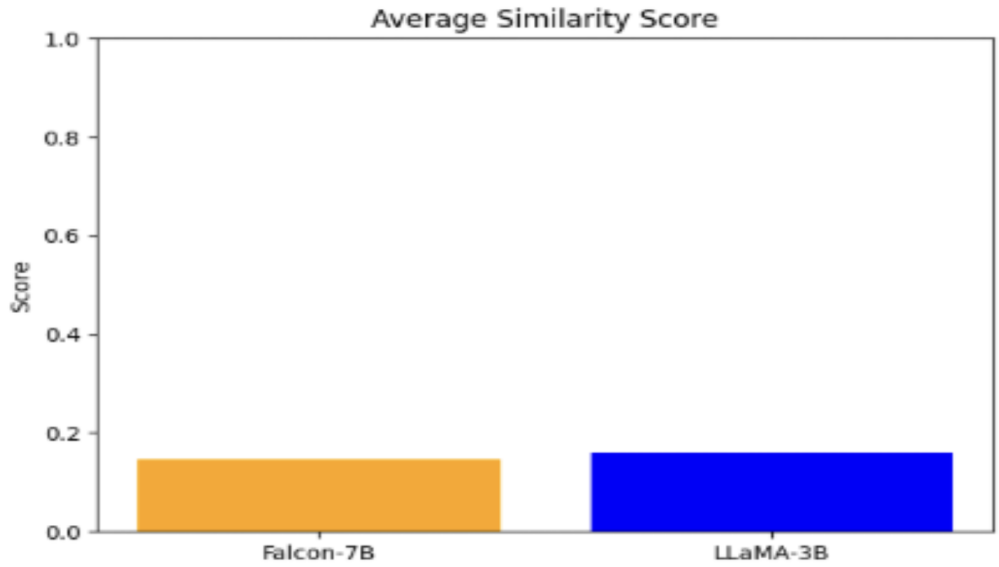

Figure 5. Comparison of Average Similarity Scores between Falcon-7B and LLaMA-3B in RAG-based QA Chatbot.

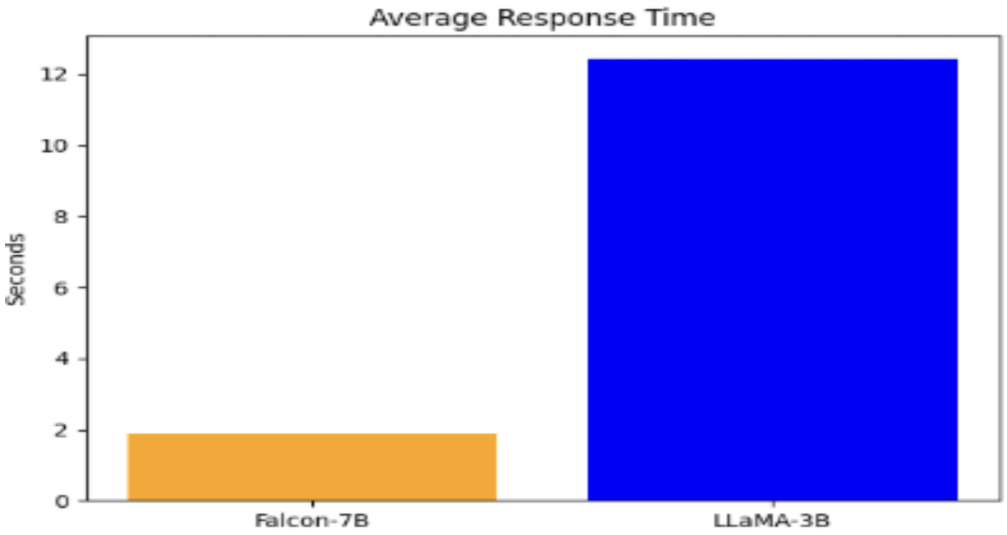

Figure 6. Average Response Time Comparison between Falcon-7B and LLaMA-3B Models.

 doi:10.20944/preprints202509.1917.v1

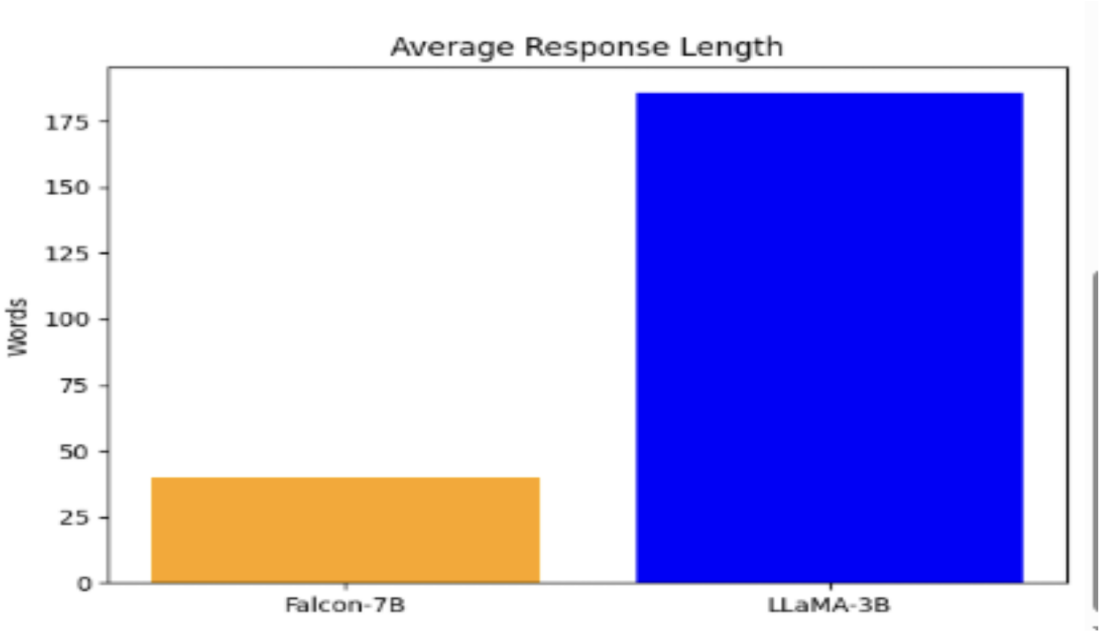

Figure 7. Average Response Length Comparison between Falcon-7B and LLaMA-3B Models.

Description While both implementations effectively demonstrate the benefits of RAG for enhancing chatbot performance, the LLaMA 3.2B with LoRA offers a more practical and efficient solution for real-world deployment where compute and latency are constrained.

The Falcon-7B, on the other hand, remains a strong choice for scenarios where model capacity and generation depth are prioritized.

4.3. Conclusion and Future Scope

This work highlights the development of an efficient Retrieval-Augmented Generation (RAG) based QA chatbot builder leveraging LLaMA 3.2B with Low-Rank Adaptation (LoRA). The integration of RAG ensures that the model generates contextually accurate and knowledge-grounded responses, while LoRA enables lightweight fine-tuning that significantly reduces computational cost and training overhead. By combining retrieval with generation, the system achieves better factual accuracy, scalability, and adaptability across diverse domains compared to conventional large-scale fine-tuning approaches. The proposed framework demonstrates that smaller language models, when efficiently optimized, can deliver performance comparable to larger models, making them practical for real-world deployment where resources are constrained.

Despite these contributions, several avenues remain open for exploration. Future work could focus on enhancing retrieval quality through hybrid dense-sparse indexing mechanisms and knowledge graph integration for structured reasoning. Incorporating multi-modal retrieval capabilities would further broaden applicability to domains requiring images, audio, or video references. Additionally, extending LoRA-based fine-tuning to continual learning scenarios would enable the chatbot to dynamically adapt to evolving knowledge without catastrophic forgetting. Exploring reinforcement learning from human feedback (RLHF) could refine response quality and alignment with user expectations. Finally, comprehensive benchmarking across domains such as healthcare, education, and customer service will validate the framework's generalizability and robustness. These directions promise to elevate the efficiency, interpretability, and real-world impact of RAG-based chatbot systems.

References

1. Barzilai, G., & Ferraris, S. D. (2023). Developing a Data Classification Framework. In S. D. Ferraris (Ed.), The Role of Prototypes in Design Research: Overview and Case Studies (pp. 9–37). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-24549-7_2

 doi:10.20944/preprints202509.1917.v1

2. Brynjolfsson, E., Li, D., & Raymond, L. R. (2023). Generative AI at work. National Bureau of Economic Research.
3. Carvalho, I., & Ivanov, S. (2023). ChatGPT for tourism: Applications, benefits and risks. Tourism Review, 79(2), 290–303. https://doi.org/10.1108/TR-02-2023-0088
4. Chaturvedi, R., & Verma, S. (2023). Opportunities and Challenges of AI-Driven Customer Service. In J. N. Sheth, V. Jain, E. Mogaji, & A. Ambika (Eds.), Artificial Intelligence in Customer Service: The Next Frontier for Personalized Engagement (pp. 33–71). Springer International Publishing. https://doi.org/10.1007/978-3-031-33898-4_3
5. Chen, X., & Beaver, I. (2022). An Adaptive Deep Clustering Pipeline to Inform Text Labeling at Scale (arXiv:2202.01211). arXiv. https://doi.org/10.48550/arXiv.2202.01211
6. Dasgupta, I., Lampinen, A. K., Chan, S. C. Y., Sheahan, H. R., Creswell, A., Kumaran, D., McClelland, J. L., & Hill, F. (2023). Language models show human-like content effects on reasoning tasks (arXiv:2207.07051). arXiv. http://arxiv.org/abs/2207.07051
7. Doellgast, V., O'Brady, S., Kim, J., & Walters, D. (2023). AI in Contact Centers: Artificial Intelligence and Algorhythmic Management in Frontline Service Workplaces. https://ecommons.cornell.edu/items/2c04c957-d672-400e-9aed-adb5f6ace640
8. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Guo, Q., Wang, M., & Wang, H. (2023). Retrieval-Augmented Generation for Large Language Models: A Survey (arXiv:2312.10997; Version 1). arXiv. http://arxiv.org/abs/2312.10997
9. Herterich, M. M., Dremel, C., Wulf, J., & Vom Brocke, J. (2023). The emergence of smart service ecosystems—The role of socio-technical antecedents and affordances. Information Systems Journal, 33(3), 524–566. https://doi.org/10.1111/isj.12412
10. Iparraguirre-Villanueva, O., Obregon-Palomino, L., Pujay-Iglesias, W., Sierra-Liñan, F., & Cabanillas-Carbonell, M. (2023). Productivity of incident management with conversational bots-a review. https://repositorio.uwiener.edu.pe/handle/20.500.13053/9592
11. Kanbach, D. K., Heiduk, L., Blueher, G., Schreiter, M., & Lahmann, A. (2023). The GenAI is out of the bottle: Generative artificial intelligence from a business model innovation perspective. Review of Managerial Science. https://doi.org/10.1007/s11846-023-00696-z
12. Lin, X., Wang, W., Li, Y., Yang, S., Feng, F., Wei, Y., & Chua, T.-S. (2024). Data-efficient Fine-tuning for LLM-based Recommendation (arXiv:2401.17197). arXiv. https://doi.org/10.48550/arXiv.2401.17197
13. Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., He, H., Li, A., He, M., Liu, Z., Wu, Z., Zhao, L., Zhu, D., Li, X., Qiang, N., Shen, D., Liu, T., & Ge, B. (2023). Summary of ChatGPT-Related Research and Perspective Towards the Future of Large Language Models. Meta-Radiology, 1(2), 100017. https://doi.org/10.1016/j.metrad.2023.100017
14. OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., … Zoph, B. (2023). GPT-4 Technical Report (arXiv:2303.08774). arXiv. http://arxiv.org/abs/2303.08774
15. Reinhard, P., Li, M. M., Dickhaut, E., Peters, C., & Leimeister, J. M. (2023). Empowering Recommender Systems in ITSM: A Pipeline Reference Model for AI-Based Textual Data Quality Enrichment. In A. Gerber & R. Baskerville (Eds.), Design Science Research for a New Society: Society 5.0 (pp. 279–293). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-32808-4_18
16. Reinhard, P., Li, M. M., Peters, C., & Leimeister, J. M. (2024). Generative AI in Customer Support Services: A Framework for Augmenting the Routines of Frontline Service Employees. Customer Support Services: A Framework for Augmenting the Routines of Frontline Service Employees (January 6, 2024). Hawaii International Conference on System Sciences (HICSS), Waikiki, Hawaii, USA. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4612768
17. Reinhard, P., Wischer, D., Verlande, N., Neis, N., & Li, M. (2023). Towards designing an AI-based conversational agent for on-the-job training of customer support novices. International Conference on Design Science Research (DESRIST). https://pubs.wi-kassel.de/wp-content/uploads/2023/06/JML_931.pdf
18. Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., & Fedus, W. (2022). Emergent Abilities of Large Language Models (arXiv:2206.07682). arXiv. http://arxiv.org/abs/2206.07682

 doi:10.20944/preprints202509.1917.v1

19. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT (arXiv:2302.11382). arXiv. http://arxiv.org/abs/2302.11382
20. Singh, S. (2025). An Enhanced Large Language Model For Cross Modal Query Understanding System Using DL-KeyBERT Based CAZSSCL-MPGPT. arXiv preprint arXiv:2502.17000.
21. Wulf, J., & Meierhofer, J. (2023). Towards a Taxonomy of Large Language Model based Business Model Transformations (arXiv:2311.05288). arXiv. http://arxiv.org/abs/2311.05288