SlimRAG: Better Retrieval-Augmented Generation (RAG) for Small Language Models (SLMs)

Akshat Singhal[1] **Chenchen Wang**[2] **Devesh Maheshwari**[3] **Xi Hong**[4]
[1]Department of Computer Sciences, [2]Department of Electrical and Computer Engineering,
[3]Department of Computer Sciences , [4]Information School,
University of Wisconsin–Madison
{asinghal28, cwang922, dmaheshwar22, xihong39}@wisc.edu

1 Introduction

Although Retrieval-Augmented Generation (RAG) has achieved remarkable performance, its effectiveness diminishes significantly in resource-constrained settings, particularly when applied to small language models (SLMs). Existing RAG architectures were originally designed to exploit the advanced reasoning and comprehension abilities of large language models (LLMs). Consequently, they struggle to adapt to the inherent limitations of SLMs across several critical aspects such as multi-step reasoning and nuanced information synthesis (Fan et al., 2025). The possible loss of contextual information and weakness in capturing inter-chunk relationships also undermines their capacity to retrieve coherent and interconnected knowledge effectively (Zhang et al., 2025; Günther et al., 2025). To address these shortcomings, MiniRAG was introduced (Fan et al., 2025), redefining RAG for SLMs through two core innovations: (1) heterogeneous graph indexing and (2) lightweight graph-based knowledge retrieval. Nevertheless, despite these advancements, the broader potential of applying RAG to SLMs remains largely unexplored.

Our study aims to optimize RAG applications on SLMs by developing a novel RAG system, **SlimRAG**, based on MiniRAG. Specifically, we will:

(1) Provides empirical insights into adapting RAG architectures for SLMs by incorporating several techniques from advanced RAG systems (e.g., post-retrieval enhancements, evaluation feedback mechanism) to improve retrieval and generation performance.

(2) Offers a benchmark and design framework for future research on lightweight and graph-based RAG systems.

2 Literature review

RAG for LLMs: A Survey (Gao et al., 2024) Discusses challenges with directly using LLM without RAG: hallucinations, outdated knowledge, and untraceable reasoning process. Three RAG frameworks are discussed: Naive, Advanced, and Modular RAG. Naive RAG is a simple indexing of text chunks into a database, retrieving top K chunks based on semantic similarity with the user query, and finally generating a response via LLM. The second framework, Advanced RAG, introduces some pre-retrieval strategies, namely query rewrite and transformation, add metadata with text chunks, and alignment optimization. Post-retrieval stage includes reranking text chunks, context compression, and filtering of noise in context. Modular RAG introduces several modules like search, memory, routing, and predict module, each handling a specialised task in different stages. There are multiple augmentation techniques discussed, like an iterative self-feedback loop in the architecture, where LLM will rate its own generation and make changes to the query based on that feedback. There is a mention of SLMs in the paper where SLMs detect and remove noisy tokens, making the text less understandable to humans, but very well processed by LLMs.

MiniRAG (Fan et al., 2025) MiniRAG introduces an innovative retrieval-augmented generation (RAG) framework designed specifically for small language models (SLMs). Unlike traditional LLM-oriented systems including LightRAG and GraphRAG, MiniRAG reshapes the RAG pipeline around simplicity, structure, and computational efficiency. Its design is based on three key insights: SLMs excel at pattern matching rather than deep semantics, explicit structural representations can compensate for weak reasoning, and decomposing complex RAG tasks into modular steps enhances robustness. MiniRAG's architecture encompasses two innovative components: heterogeneous graph indexing, which constructs a rich semantic network linking text chunks and entities, and lightweight graph-based knowledge retrieval, which maps queries to graph nodes and go through semantic paths to retrieve answers efficiently. The system is evaluated on synthetic daily communication data and multi-hop short-document datasets. Through this architecture, MiniRAG achieves efficient and coherent retrieval in environments where computational resources are highly constrained.

LightRAG (Guo et al., 2025) introduces an efficient graph-based approach to Retrieval-Augmented Generation that addresses the high cost and slow processing of prior models like Naive RAG and community-based traversal methods used in GraphRAG. Its indexing process is built on three key functions — entity-relationship extraction, LLM-based key–value profiling, and de-duplication — creating a compact, knowledge-rich graph. For retrieval, LightRAG employs a dual-view paradigm with local (fine-grained) retrieval for factual queries and global (conceptual) retrieval for holistic understanding. This structure allows it to capture both detail and context efficiently. Moreover, LightRAG supports incremental updates and drastically reduces token usage, making it a lightweight yet powerful alternative to traditional and graph-based RAG systems.

GraphRAG (Han et al., 2025) represents an emerging direction in Retrieval-Augmented Generation that integrates graph-structured knowledge into retrieval and reasoning. Unlike text-only RAG frameworks, GraphRAG explicitly models entities and their relationships through nodes and edges, enabling the capture of multi-hop dependencies and contextual links often lost in flat text. A typical pipeline maps a query to relevant nodes or subgraphs in a knowledge graph, applies graph traversal or GNNs to extract a coherent subgraph, and converts it into a textual form for language model generation. This graph-based reasoning improves interpretability, factual grounding, and multi-hop question answering while reducing hallucination.

3 Proposed Methodology

Our proposed method focuses on enhancing the post-retrieval stage of the MiniRAG pipeline to better suit the constraints of Small Language Models (SLMs). The approach integrates three key components: a fine-tuned rerank module, a context compression module, and an iterative SLM-as-a-judge pipeline. Once the text chunks are retrieved from the retriever, we will integrate these approaches.

First, the rerank module is fine-tuned to accurately identify and prioritize the most semantically relevant documents for a given query. We will use positive and negative pairs with the user query to finetune the rerank model using contrastive loss. We will organize the retrieved chunks based on their similarity scores (given by the rerank model) with user queries. By optimizing document ranking based on contextual alignment rather than surface-level similarity, this component ensures that the most informative and pertinent evidence is prioritized to be given to SLM for the generation of the final answer.

Next, the context compression module condenses the retrieved text into concise, information-rich representations. This step also eliminates redundancy and noise while retaining the key facts, thereby reducing the token footprint and improving efficiency, an essential consideration when working with SLMs that have limited context windows and hallucinate if redundant content is given as input (Gu et al., 2025). We will experiment with different summarizer language models and SLMs to build this module.

Finally, the iterative SLM-as-a-judge pipeline introduces a lightweight self-verification and refinement mechanism. In this stage, the SLM not only evaluates its generated response for factual consistency, coherence, and contextual grounding against the retrieved documents but also uses this feedback to rewrite the user query for improved retrieval. The updated query is then passed back to the retriever to fetch enhanced text chunks that better align with the user's intent. This iterative process progressively refines both the retrieved evidence and the generated answer, resulting in more accurate, focused, and well-grounded outputs. To prevent indefinite feedback cycles, the loop terminates once the judge deems the response satisfactory or when a predefined quality threshold or maximum iteration limit is reached. By continuously closing the loop between generation and retrieval within these controlled bounds, the system significantly enhances reliability while minimizing hallucinations and irrelevant responses in the final output.

Together, these components optimize retrieval quality, reduce contextual noise, and strengthen factual grounding, enabling our RAG architecture to effectively address the computational and contextual limitations inherent to SLMs while maintaining high-quality generation.

Example of effectiveness:

Retrieved Context: LiHua: Hey Adam! Just wanted to ask for the *Wi-Fi password*. Also, are there any *house rules* I should know about? *Adam Smith*: Sure! The Wi-Fi password is "Family123". For house rules, please keep noise to a minimum during late hours and take good care of the property. *User Question:* What were the house rules discussed in above conversation?
LLM response: The house rules discussed were: Keep noise to a minimum during late hours. Take good care of the property.
SLM response: Based on the dialogue provided: 1. **Wi-Fi Password**: The Wi-Fi password was set as "Family123".2. **Noise Control**: The speaker advised that noise levels should be kept low at night during late hours.

After using the context compression module, we will be able to reduce the noise from the retrieved context pertaining to "Wi-Fi password" in above example.

4 Experiments

4.1 Datasets and Evaluation Metrics

This project adopts the **LiHua-World** dataset, a text corpus built for local Retrieval-Augmented Generation (RAG) scenarios. The dataset contains one year of chat records (January–December 2026) from a virtual user named *LiHua*, covering diverse aspects of daily life such as social interactions, fitness, entertainment, and household activities. Each record includes timestamps, sender identities, and message types, providing rich temporal and contextual information for retrieval-based reasoning.

Model performance is measured by two quantitative metrics. Accuracy reflects the proportion of correctly generated answers that match or semantically align with reference answers, while the error rate measures the share of incorrect responses. The evaluation of generated answers will be conducted by an external large language model. Specifically, we employ the ChatGPT-4o model with carefully designed prompts to assess the quality and correctness of the model outputs.

4.2 Computational Estimation

All experiments are conducted on a single NVIDIA A100 GPU with 40 GB of VRAM. Before launching the main experiments, we carried out a preliminary run to reproduce the MiniRAG pipeline using one-quarter of the LiHua-World dataset and the `Qwen2.5-3B-Instruct` model. The reproduced results are consistent with those reported in previous MiniRAG studies, confirming the reliability of our experimental setup. In this preliminary experiment, both the indexing and inference process required approximately five hours to complete.

4.3 Model Checkpoints and Codebase

The SlimRAG framework integrates several small instruction-tuned language models, including `Qwen2.5-3B-Instruct`, `LLaMA-3.3-3B`, and `Phi-3.5-mini-instruct`, all of which are implemented through the HuggingFace Transformers framework for consistent API and reproducibility. The main code framework of this project will be based on the open source code repository of LightRAG and MiniRAG

Research value and plan

While small language models (SLMs) have shown remarkable progress in language understanding and multimodal reasoning, their potential in retrieval-augmented generation (RAG) tasks remains underexplored. By developing a new RAG system **SlimRAG**, this research aims to illuminate the future of applying RAG in SLMs through:

(1) A systematic benchmark evaluating SlimRAG performance across SLMs, comparing Naïve RAG, Graph RAG, Light RAG, and MiniRAG, thereby addressing a key gap where most RAG studies focus on large-scale models.

(2) The development of a more accurate computing-optimized question-answering system.

Work will be equitably distributed among team members as follows:

Literature Review: Akshat, Chenchen, Devesh, and Xi will jointly review and discuss relevant studies, with Xi leading discussions and summarization.

Methodology & Experiment Design: All members will collaboratively design the methodology and experiments, with Akshat and Devesh leading the discussions.

Experiment Execution: All members will participate in running experiments, led by Chenchen.

Manuscript Preparation: All members will contribute to drafting and refining the manuscript.

References

Tianyu Fan, Jingyuan Wang, Xubin Ren, and Chao Huang. 2025. Minirag: Towards extremely simple retrieval-augmented generation. *Preprint*, arXiv:2501.06713.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2025. A survey on llm-as-a-judge. *Preprint*, arXiv:2411.15594.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. Lightrag: Simple and fast retrieval-augmented generation. *Preprint*, arXiv:2410.05779.

Michael Günther, Isabelle Mohr, Daniel James Williams, Bo Wang, and Han Xiao. 2025. Late chunking: Contextual chunk embeddings using long-context embedding models. *Preprint*, arXiv:2409.04701.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A. Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qi He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. 2025. Retrieval-augmented generation with graphs (graphrag). *Preprint*, arXiv:2501.00309.

Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Hao Chen, Yilin Xiao, Chuang Zhou, Junnan Dong, Yi Chang, and Xiao Huang. 2025. A survey of graph retrieval-augmented generation for customized large language models. *Preprint*, arXiv:2501.13958.