A Comprehensive Study on Retrieval Augmented Generation Methods for More Robust LLMs

Andrea Palmieri
Artificial Intelligence For Science and Technology
Università degli Studi di Milano-Bicocca
palmieri.andrea2000@gmail.com

A thesis submitted in fulfillment of the requirements for the degree of

Master of Science
in
Artificial Intelligence For Science and Technology

Supervisor: Dr. Napoletano Paolo

June 5, 2025

A Comprehensive Study on Retrieval Augmented Generation Methods for More Robust LLMs

Andrea Palmieri
Artificial Intelligence For Science and Technology
Università degli Studi di Milano-Bicocca
palmieri.andrea2000@gmail.com

June 5, 2025

Abstract

This thesis investigates various methods to enhance the robustness of Large Language Models (LLMs) through Retrieval Augmented Generation (RAG). It explores fundamental RAG concepts, advanced retrieval and optimization techniques, prompt engineering strategies, and the impact of different LLMs. A key part of this work is an experimental evaluation of retrieval system performance, focusing on precision and recall, comparing baseline embedding models with novel thresholding techniques and reranking approaches.

Contents

Chapter 1

Introduction

This thesis delves into the domain of Retrieval Augmented Generation (RAG), a paradigm aimed at enhancing the capabilities of Large Language Models (LLMs) by grounding their responses in external knowledge sources. While LLMs have demonstrated remarkable proficiency in various natural language tasks, they are often susceptible to issues like hallucination and reliance on outdated internal knowledge [1]. [1] RAG seeks to mitigate these limitations by integrating a retrieval step that fetches relevant information from a corpus, which is then used by the LLM to generate more accurate, factual, and contextually appropriate responses [2]. [2]

This study provides a comprehensive overview of RAG, exploring its foundational components, advanced optimization techniques, and evaluation methodologies. We investigate various aspects, from chunking strategies and embedding models to sophisticated reranking mechanisms and dynamic thresholding for similarity scores. Furthermore, the role of prompt engineering and the selection of LLMs are examined for their impact on overall RAG system performance.

Chapter 2

Fundamentals of Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) systems combine the strengths of pre-trained language models with information retrieval systems [2]. [2] This chapter lays the groundwork by explaining the core mechanics of RAG, the primary challenges in its implementation, the role of vector databases, and strategies for mitigating common issues like LLM hallucinations.

2.1 How it Works

At its core, a RAG system operates in two main stages: retrieval and generation. When a query is posed, the retriever first searches a knowledge base (e.g., a collection of documents, a database) for information relevant to the query. This retrieved context is then passed, along with the original query, to a language model, which generates the final response.

2.2 Main Challenges

Several challenges need to be addressed for effective RAG systems, including:

- Ensuring the relevance and quality of retrieved documents.
- Optimizing the trade-off between retrieval speed and comprehensiveness.
- Handling noisy or conflicting information in the retrieved context.
- Seamlessly integrating the retrieved context into the generation process.

2.3 Vector Databases

Vector databases are crucial for efficient similarity search in RAG. They store embeddings of text chunks and allow for fast retrieval of the most similar chunks to a query embedding.

2.4 Mitigating Hallucinations

By providing relevant, factual context, RAG significantly reduces the tendency of LLMs to hallucinate or generate factually incorrect statements [3]. [3] The quality of the retrieved context is paramount for this.

Chapter 3

Retrieval and Optimization Methods

The effectiveness of a RAG system heavily depends on the quality of its retrieval component and the optimization techniques employed. This chapter explores various methods to enhance information retrieval for RAG.

3.1 Chunking Techniques: Late Chunking vs. Naive Chunking

Chunking, the process of breaking down large documents into smaller, manageable pieces, is a critical preprocessing step. We will discuss different strategies like naive fixed-size chunking versus more adaptive methods sometimes referred to as "late chunking" or semantic chunking.

3.2 Different Embedding Models

The choice of embedding model (e.g., ada-002, Sentence-BERT, custom models) significantly impacts the semantic representation of text and thus the relevance of retrieved chunks.

3.3 Contextual Embeddings

Contextual embeddings, which capture the meaning of words in their specific context, are generally preferred over static word embeddings for RAG.

3.4 BM25 and TF-IDF for Reranking

Traditional IR methods like BM25 and TF-IDF can be effectively used as a reranking step to refine the results from an initial dense retrieval [4]. [4] These methods focus on lexical overlap and term importance.

3.5 Hybrid Systems: Combining Similarity with BM25/TF-IDF for Reranking

Hybrid systems aim to leverage the strengths of both dense retrieval (semantic similarity) and sparse retrieval (keyword matching like BM25) for improved reranking performance [5]. [5]

3.6 Dynamic Similarity Thresholding

Instead of using a fixed similarity threshold for retrieved chunks, dynamic thresholding adapts the threshold based on the query or the distribution of similarity scores [6]. [6] This can help in retrieving a more appropriate number of relevant chunks.

Chapter 4

Prompt Engineering

Prompt engineering plays a vital role in guiding the LLM to effectively utilize the retrieved context and generate the desired output in a RAG system.

4.1 Concepts of Prompt Engineering

This section will cover the fundamental principles of designing effective prompts for RAG, including how to structure the prompt to include the query and the retrieved context.

4.2 Prompt Styles

Different styles of prompts (e.g., zero-shot, few-shot, instruction-based) can be used depending on the task and the LLM.

4.3 Prompt Evaluation using Golden Datasets and LLM-as-a-Judge (e.g., DeepEval)

Evaluating the quality of prompts is crucial. This section will discuss methods using golden datasets and leveraging other LLMs as evaluators (LLM-as-a-Judge) with frameworks like DeepEval [7]. [7]

4.4 Evaluation Metrics: GEval, DAG, Hallucination, Answer Relevancy, Faithfulness

Key metrics for evaluating RAG outputs, often assessed by LLM judges, include G-Eval, Answer Relevancy, Faithfulness, and specific metrics for hallucination detection [8]. [8]

Chapter 5

Comparative Analysis of Different LLMs

The choice of the Large Language Model is a critical factor in the performance of a RAG system.

5.1 Performance Evaluation based on Model and Context Window Size

This section will discuss how different LLMs (e.g., GPT-3.5, GPT-4, Llama series, etc.) perform in a RAG setup. The impact of their context window size on handling retrieved information will also be analyzed. Different models may have varying strengths in synthesizing information, adhering to context, and avoiding hallucinations.

Chapter 6

Identifying Relevant Chunks for Responses

Understanding which retrieved chunks contribute most to the LLM's final answer is important for debugging, improving the retriever, and providing transparency.

6.1 Analyzing Chunk-Response Alignment

This section will explore methods to analyze the alignment between the generated response and the specific chunks of retrieved context that were used. This can involve techniques like attention analysis (if applicable) or other methods to trace information flow.

Chapter 7

Experiments and Results

This chapter details the experimental setup and presents the results of the performance investigation into the retrieval system. The primary goal is to evaluate different strategies for improving precision and recall in a RAG pipeline.

7.1 Experimental Setup

7.1.1 Dataset

Describe the dataset used for the experiments (e.g., a question-answering dataset, a domain-specific document collection).

7.1.2 Baseline System

The baseline retrieval system uses the `ada-002` embedding model. A predefined function is applied to the similarity scores to implement a fixed threshold for relevance.

7.1.3 Evaluation Metrics

The primary metrics for evaluating the retrieval component are:

- **Precision@k**: The proportion of retrieved chunks in the top k that are relevant.
- **Recall@k**: The proportion of all relevant chunks in the dataset that are retrieved in the top k.
- **F1-score@k**: The harmonic mean of Precision@k and Recall@k.

These metrics are crucial for understanding the effectiveness of the retrieval stage in RAG systems [9]. [9]

7.2 Performance Investigation of Retrieval System

This section focuses on the comparative performance analysis as outlined in the introduction.

7.2.1 Method 1: Baseline (ada-002 with Fixed Threshold)

Detail the performance of the baseline system using the `ada-002` embedding model and a standard, fixed thresholding function on the cosine similarity scores.

7.2.2 Method 2: Alternative Thresholding Strategies with ada-002

Investigate different methods for computing or applying a threshold to the similarity scores from `ada-002`. This could include:

- **Dynamic Thresholding**: Adapting the threshold based on query characteristics or score distributions (e.g., using Kernel Density Estimation as described by [6]). [6]
- **Top-N Cutoff**: Simply taking the top N most similar chunks, varying N.
- Other statistical approaches to determine an optimal cutoff.

The original similarity function (e.g., cosine similarity from ada-002) remains the same, but the method of deciding which chunks are "relevant enough" changes.

7.2.3 Method 3: Reranking Model with Original Thresholding Function

Here, the initial set of chunks is retrieved using `ada-002` (potentially a larger initial set). A reranking model (e.g., a cross-encoder based on BERT or T5, as discussed in reranking surveys [5]) [5] is then applied to these retrieved chunks to re-calculate similarity scores. The original thresholding function is then applied to these *new* reranked scores.

7.2.4 Method 4: Reranking Model with Alternative Thresholding Strategies

This combines the reranking approach from Method 3 with the alternative thresholding strategies from Method 2. The reranking model provides new similarity scores, and then dynamic or other advanced thresholding methods are applied to these reranked scores.

7.3 Results and Discussion

Present the precision, recall, and F1-score results for each method. Include tables and potentially graphs to compare the performance. Discuss the implications of the results, highlighting which strategies offer the best improvements over the baseline and under what conditions.

Chapter 8

Conclusions and Future Work

This thesis has provided a comprehensive study of Retrieval Augmented Generation methods for enhancing the robustness of Large Language Models.

8.1 Future Work

For example, exploring more advanced reranking models or adaptive chunking strategies could yield further improvements. Investigating the energy efficiency of different RAG pipelines is also an emerging area of interest.

Bibliography

[1] Gao Chen and et al. "A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions". In: *arXiv preprint arXiv:2405.08995* (2024). URL: https://arxiv.org/abs/2405.08995.

[2] Anonymous Authors. "RAG and RAU: A Survey on Retrieval-Augmented Language Model in Natural Language Processing". In: *Papers With Code* (2024). Accessed: June 5, 2025. Search result [1]. URL: https://paperswithcode.com/paper/rag-and-rau-a-survey-on-retrieval-augmented.

[3] Prompt Engineering Guide. *Retrieval Augmented Generation (RAG) for LLMs - Prompt Engineering Guide*. Accessed: June 5, 2025. Based on search result [2]. 2023. URL: https://www.promptingguide.ai/techniques/rag.

[4] Yutao Zhu et al. "Large Language Models for Information Retrieval: A Survey". In: *arXiv preprint arXiv:2303.17105* (2024). Based on search result [1] from 'reranking models for information retrieval survey'. URL: https://arxiv.org/abs/2303.17105.

[5] Kaouthar Djoudi, Zaia Alimazighi, and Badiâa Dellal Hedjazi. "Information retrieval with query expansion and re-ranking: a survey". In: *IET Software* (2025). Based on search result [3] from 'reranking models for information retrieval survey'. URL: https://www.researchgate.net/publication/380108341_Information_retrieval_with_query_expansion_and_re-ranking_a_survey.

[6] Dell Technologies Info Hub. *Dynamic thresholding | Product Support Quick Notes Retrieval*. Accessed: June 5, 2025. Based on search result [1] from 'dynamic similarity thresholding information retrieval'. 2023. URL: https://www.dell.com/support/kbdoc/en-us/000214997/dynamic-thresholding-product-support-quick-notes-retrieval.

[7] Qdrant. *Best Practices in RAG Evaluation: A Comprehensive Guide*. Accessed: June 5, 2025. Based on search result [1] from 'evaluating RAG systems precision recall'. 2024. URL: https://qdrant.tech/articles/rag-evaluation/.

[8] Pinecone. *RAG Evaluation: Don't let customers tell you first.* Accessed: June 5, 2025. Based on search result [2] from 'evaluating RAG systems precision recall'. 2024. URL: https://www.pinecone.io/learn/rag-evaluation/.

[9] RidgeRun.ai. *How to Evaluate Retrieval Augmented Generation (RAG) Systems.* Accessed: June 5, 2025. Based on search result [3] from 'evaluating RAG systems precision recall'. 2024. URL: https://ridgerun.ai/blog/how-to-evaluate-retrieval-augmented-generation-rag-systems/.