Development of an AI-Powered Question Answering System for PDF and Web Content using Retrieval-Augmented Generation

Master Thesis

by

Ketan Kishor Darekar

Matriculation number: 11037367

2025-20-06

SRH University Heidelberg

School of Information, Media and Design

Degree course: MSc Applied Computer Science

Major field: Business Computing

First Supervisor	Prof. Dr. Gerd Moeckel
Second Supervisor	Paul Tanzer

Abstract

I would

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.

I am aware that the thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the thesis as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future theses submitted. Further rights of reproduction and usage, however, are not granted here.

This paper was not previously presented to another examination board and has not been published.

_____	_____
first and last name	city, date and signature

Acknowledgements

I’m deeply thankful to Prof. Dr. Gerd Moeckel and Paul Tanzer for their invaluable guidance, advice, and valuable support throughout the research and development of this thesis. Their experience, expertise and encouragement have been influenced me to understanding the topic in depth and completing my work.

I’m also thankful to my family and friends, whose encouragement, patience and support during this journey. Your belief in me has motivated me to approach this research with full of dedication. I’m forever grateful to your love and support

Table of Contents

Table of Figures

No table of figures entries found.

Figures numbered with Roman numerals (II, III, IV, etc.) are part of the Appendix

List of Tables

List of Abbreviations

- LLM:
- NLP:
- RAG:
- BART:
-

1 Introduction

Write here Introduction

2 Literature Review

The above section gives a clear overview of concepts and technologies that relate to Retrieval-Augmented Generation (RAG) systems. It discusses why RAG is necessary, the major components, and enabling technologies that have pushed it into existence. The explanation is meant to provide an accurate representation of RAG and move the concept into the limelight with clarity and accuracy.

2.1 Foundations of Retrieval-Augmented Generation (RAG)

In the last few years, Retrieval-Augmented Generation (RAG) has become a potent paradigm in natural language processing (NLP) that extends the fact basis and contextual appropriateness of the response of large language models (LLMs). As opposed to conventional LLMs, which rely solely on pre-trained parameters, RAG systems pull external knowledge relevant to the situation in real time and provide grounded responses. The hybrid method solves important problems such as hallucinations and outdated data. This chapter outlines how and why RAG is, depicts its original organization, tells the migration from open-domain to domain-specific use, and answers the most crucial factors that determine how RAG systems work.

2.1.1 Understanding RAG: Concept and Origins

Large language models have advanced significantly in their ability to recognize and generate human language. But, at the end of the day, they are still constrained by the data they were trained with. This becomes a major issue when they are required to answer questions on fresh themes, highly technical issues, or anything that is outside of what they have seen before. Retrieval-Augmented Generation (RAG) was developed to address this limitation by allowing models to query outside of their internal knowledge store. Instead of relying exclusively on what the model has memorized, RAG includes a retrieval unit that searches other sources (papers, journals, or databases) and, depending on the content received, provides a more realistic and contextually relevant answer (Lewis et al., 2020).

With increasing interest in RAG, researchers have begun investigating how it can be tailored for purposes. Perhaps the greatest benefit of RAG is that both its retrieval and generation components can be modified separately, making it simple to change the system based on whether it needs to process a particular kind of information. Some of these uses are general-purpose and some are specific industries such as healthcare or finance. The versatility of RAG has made it a viable solution for more real-world, real-case applications. This is very helpful for my thesis because it will allow me to adapt the system in order to pull structured data out of PDFs and web pages, which will result in more precise and accurate results (Jing et al., 2024).

2.1.2 The Hallucination Problem in LLMs

Large language models tend to generate factually incorrect but coherent responses, a challenge known as the "hallucination problem." It occurs when the model produces material that is not based in fact but sounds confident and well-formed. Hallucinations often occur when the model has no access to current or domain knowledge and uses memorized patterns in pretraining. The problem is specifically vivid in areas where fact accuracy is required, such as medicine, jurisprudence, or academic research. As a countermeasure, methods must be implemented to limit model reliance on internal approximations and facilitate anchoring in verified external sources. Algorithms for retrieval augmentation ensure fewer hallucinations by enabling models to reference and leverage important papers at generation time, thereby rendering responses contextually accurate and traceable (Gupta et al., 2024).

2.1.3 From Open-Domain QA to Specialized Applications

When it first started growing, Retrieval-Augmented Generation (RAG) was primarily applied to open-domain question answering, in which the model had to access a huge and generic database such as Wikipedia. These models were generalist and could handle a wide range of problems but would not be specialized in one specific domain. But as RAG evolved, it was understood that most real-world use cases demanded the system to operate in expert domains where context, accuracy, and language are important. This evolution encouraged interest in the application of

RAG for more domain-specific uses, including in medicine, legal editing, and enterprise document processing (Gao et al., 2023).

To enable RAG to work better in real life scenarios, particularly when handling technical or formal documents, researchers have begun adapting the partitioning of the information prior to searching it. Instead of dividing texts into blocks of a uniform size, newer methods divide them according to meaning i.e., paragraphs or logical sections, so that the information that is retrieved will make sense contextually. This is especially helpful when dealing with structured documents like research papers, manuals, or legal files. Furthermore, by regulating the amount of the content the system ingests during one pass, RAG can focus solely on the most relevant parts. These are truly valuable contributions to projects like mine, which are trying to build a system that will accept questions and answer them intelligently and accurately based on lengthy PDF documents or web-based content (Pan et al., 2023).

2.2 Core Architecture and Components of RAG

This section provides the overview of a RAG system — retriever, generator, and re-ranker and how these parts interact to retrieve information, generate answers, and improve response quality. It identifies how each element makes RAG systems better, especially for complex or domain-related tasks.

2.2.1 The Retriever: Dense, Sparse, and Hybrid Models

In a RAG system, the first thing to do is discover information pertinent to answering a question. The retriever scans a great number of documents to choose the most appropriate. Traditional retrievers, or sparse retrievers, match queries with documents based on words. They are easy and quick, but they tend to lose more subtle relationships between words. Dense retrievers build upon this by representing both the query and documents as numerical vectors so that the system can identify matches based on meaning even when words don't exactly match. This makes dense retrievers more effective in applications where context matters more than exact phrasing (Shan & Shan, 2025; Tanyildiz et al., 2025).

Hybrid retrieval, which combines sparse and dense approaches, is utilized by most systems to achieve additional efficiency. The method performs well on both single keyword matching and fetching larger semantic context. Hybrid methods are particularly convenient to deal with complex content like scholarly papers or semi-structured web pages, where parts of the content are less suitable for keyword matching than others for meaning-based search. LangChain makes it easy to develop systems like these, and libraries such as FAISS, PGVector, and Pinecone see heavy utilization for optimized storage and search of data. In my own work, whether attempting to scrape data from web pages or PDFs, selecting the best method of scraping or even combining a few of them can spell the difference in the output (Vasilios Mavroudis, 2024; Yuan et al., 2024)

2.2.2 The Generator: LLMs and Response Formulation

After the retriever has identified useful information, the generator comes in and creates a proper response. Instead of constructing responses from memory, the model is now able to draw upon actual documents related to the subject. This enables it to produce seamless responses based on actual content. The generator scans the question, and the text retrieved before creating a full response. Models like BART or T5 are regularly used for the same because they are created to process input-output text tasks in a sequential manner. As the recovered text directly affects the output, the results are more accurate and relevant, particularly in those cases involving an expert or technical subject. This process defines RAG models from standard language models that depend on already known knowledge (Lewis et al., 2020).

2.2.3 The Re-Ranker: Optimizing Output Quality

The re-ranker also enhances the response of a RAG system. It reads documents and provides answers, and it also ranks the most relevant answers to the question of the user. The process removes information that is not related so that only the most valid, coherent, and useful data is prioritized. In case numerous articles appear to be similar, the re-ranker can identify which one is more connected to the topic. It is a final check that brings quality and confidence to the solution generated (Jing et al., 2024).

Depending upon how the system is configured, re-ranking can be performed before or after the response generation to make the output better. It works by comparing the content with the original question and selecting what best suits. Tools like LangChain and others make it easier to utilize this technique. This stage is especially useful when handling long or complicated texts where all of it may not be equally beneficial. With another filtering of the results, the system will stand a greater likelihood of coming up with a definite and important response (Vasilios Mavroudis, 2024).

2.3 Enhancing Retrieval and Understanding with Metadata and Chunking

Semantic chunking and metadata approaches enhance information retrieval accuracy in RAG systems. These approaches enhance the understanding and ability of the system to incorporate large amounts of text, particularly for structured texts. In this section I will describe how breaking the content up into meaningful portions and applying metadata (for filtering or ranking) leads to more focused and useful responses. I will also examine leveraging both these two techniques to enhance retrieval effectiveness for domain-specific applications.

2.3.1 Semantic Chunking Strategies

The text can be split into similarly sized chunks When the papers are large, just dividing up the text is not enough. Fixed length segments might lose important context or separate relevant documents leading to the difficulties in the matching. Semantic chunking, i.e., focusing on meaning rather than size, is an efficient strategy in RAG systems for solving this problem. Clustering related concepts improves the model to identify and extract pertinent information better. This proves to be especially useful for advanced writing, such as research papers or tutorials, where accurate ordering and organization are needed (Gao et al., 2023).

The present studies also considered the role of using chunking strategies on overall performance. Instead of just following a fixed formula, computers now fine-tune the size of the chunk and the amount of overlap, using information about document type and workload. You are free to keep longer paragraphs unchanged and not to cluster

the shorter ones to serve as adequate context. This provides the retriever more freedom to be more responsive and provides the generator a capability to come up with more coherent responses (Pan et al., 2023).

2.3.2 Metadata Extraction and Usage

Metadata is essential in RAG systems to sort and screen information. Metadata can contain author names, publication date, document type, and section headings, etc. These tags do more than given structure; they also assist the retriever in rapidly restricting the most relevant parts of a dataset. For example, if a person wants a specific piece of legislation or research study, metadata will help them locate it without having to search through worthless information. This increases the efficiency and accuracy in retrieval (Rodrigues & Branco, 2025).

To be able to utilize metadata to its fullest, newer frameworks such as LangChain provide the ability for developers to append it to document chunks during processing time. You can then use these tags to filter, score and re-rank results. The problem, however, lies in how to extract metadata in the right way and especially for PDFs or complex web pages that has non-standard layout. Rule-based systems, machine learning models trained on common patterns. If properly extracted, metadata performs the function of a map, leading the retrieval system to more effectively understand the information and provide more precise answers (Vasilios Mavroudis, 2024).

2.3.3 Chunk-Metadata Synergy

Combination of metadata and chunking mechanisms is necessary in order to make retrieval smart and pertinent. Pre-processing of text to chunks of metadata like author, headings, or timestamp allows for systems to create content which is semantically equivalent and contextually equivalent. Enabling filtering and ranking of results, mostly domain configurations where precision might not necessarily be at the center. This is achieved through projects like LangChain, which assigns ordered labels to every article, providing smarter and more accurate responses to questions (Rodrigues & Branco, 2025; Vasilios Mavroudis, 2024).

2.4 Vector Embeddings and Semantic Indexing

Building RAG systems for effectiveness, it is essential to understand how information is stored and represented. The subject matter involves how language is embedded into numerical vectors where machines can potentially recognize beyond word meaning. It also refers to vector databases such as FAISS and PGVector, where such embeddings are stored and how indexing methods facilitate efficient and accurate retrieval of key information. All these technologies add retrieval speed, smarts, and scalability.

2.4.1 Introduction to Embeddings in NLP

In order to correctly process language, machines must convert words and phrases into numbers. In this situation, embeddings will come in handy. Rather than focusing on phrases, embeddings describe the meaning of phrases by placing similar words close to one another in a more-than-two-dimensional space. It enables systems to appreciate that "doctor" and "nurse" are close to each other despite the difference in their letters. Such vector depictions are currently the foundation of modern NLP and form the core of RAG systems for them to learn and parse information in an informative way (Wang et al., 2022).

2.4.2 Vector Databases: FAISS, PGVector, Pinecone

Once the text is translated into embeddings, it should be stored in a manner that can be fetched and interpreted quickly. Vector databases facilitate that. FAISS, PGVector, and Pinecone are libraries meant to store and query these embeddings efficiently. They enable the system to query the most semantically nearest content efficiently when queried by a user. All of the databases have varying advantages, some are quicker, some are quicker to integrate, and others can handle very large datasets—so all are valuable choices based on the project's requirements (Pan et al., 2023).

2.4.3 Indexing Techniques: HNSW, IVF, Flat

When working with vector embeddings in large data, getting the information accurately can be challenging. This is where indexing techniques such as HNSW (Hierarchical Navigable Small World), IVF (Inverted File Index), and Flat indexing

come in handy. All three techniques are compromises based on a balance between speed, memory, and precision.

Flat indexing does an exact search across all vectors and is thus the most accurate but slow choice. IVF accelerates the search by clustering vectors into groups and only scanning the corresponding ones during retrieval time. HNSW builds a graph-based structure that enables very rapid searches by traveling vector neighbourhoods. These methods are important to improve retrieval task performance for applications like as RAG, which rely on response speed and relevance (Tanyildiz et al., 2025).

2.5 Conversational Memory in Multi-Turn QA

This section shows the issue of how conversational memory aids multi-turn question answering systems. This explains how computers maintain context in recurring user interactions, allowing for coherent, correct answers. It shows how user queries and previous conversations are properly linked by explaining different memory architectures, short-term and long-term. The section also covers methods such as conversation threading and memory chaining, enabling context flow to be maintained in Retrieval-Augmented Generation pipelines.

2.5.1 Short-Term vs. Long-Term Memory Architectures

Memory is an essential component in multi-turn question answering systems because it keeps the conversation in context. Short-term memory holds the recent interactions, so the system can generate reasonable responses without re-processing the whole conversation. It is quick and efficient but lacks capabilities. Long-term memory, on the other hand, enables models to retain important knowledge from previous sessions. This creates solutions that are more personalized and context specific. Both memories serve different purposes, and through their integration, performance is enhanced as well as context retention. With the development of conversational AI, the right memory approach becomes a must so that conversations become smart and seamless (Wu et al., 2025).

2.5.2 Conversation Threading and Memory Chains

Multi-turn question answering models must remember more than isolated inputs; they must remember the conversation direction as well. Memory chains and conversation threading come in handy for this. Instead of treating each user message as an isolated query, memory chains connect the messages so that the model will be aware of how the current question follows from previous ones. For example, if a user asks, "What are the benefits of solar energy?” followed by "Are there any environmental downsides?", the system will understand that the second question is still pointing to solar energy. LangChain achieves this through memory aspects that store brief summaries or complete histories of a conversation, allowing the model to reply in coherence and with knowledge. These kinds of frameworks are critical in creating smart agents that reply more naturally and sensitively in extended conversations (LangChain, 2025).

3 Methodology

Write herehvjdvh

4 Implementation

write here about implementation

5 Evaluation & Results

Write here about evaluation and results

6 Discussion

Write here about discussion

7 Future Work

write here about

8 Conclusion

Write here about conclusi

9 References

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2023). *Retrieval-Augmented Generation for Large Language Models: A Survey.* https://doi.org/10.48550/arXiv.2312.10997

Gupta, S., Ranjan, R., & Singh, S. N. (2024). *A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions.* https://doi.org/10.48550/arXiv.2410.12837

Jing, Z., Su, Y., & Han, Y. (2024). *When Large Language Models Meet Vector Databases: A Survey.* https://doi.org/10.48550/arXiv.2402.01763

LangChain. (2025). *Memory Components – LangChain*. https://python.langchain.com/api_reference/langchain/memory.html

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.* https://doi.org/10.48550/arXiv.2005.11401

Pan, J. J., Wang, J., & Li, G. (2023). *Survey of Vector Database Management Systems.* https://doi.org/10.48550/arXiv.2310.14021

Rodrigues, J., & Branco, A. (2025). Meta-prompting Optimized Retrieval-Augmented Generation. In M. F. Santos, J. Machado, P. Novais, P. Cortez, & P. M. Moreira (Eds.), *Lecture Notes in Computer Science. Progress in Artificial Intelligence* (Vol. 14969, pp. 203–214). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-73503-5_17

Shan, R., & Shan, T. (2025). Retrieval-Augmented Generation Architecture Framework: Harnessing the Power of RAG. In R. Xu, H. Chen, Y. Wu, & L.-J. Zhang (Eds.), *Lecture Notes in Computer Science. Cognitive Computing - ICCC 2024* (Vol. 15426, pp. 88–104). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-77954-1_6

Tanyildiz, D., Ayvaz, S., & Amasyali, M. F. (2025). Enhancing Retrieval-Augmented Generation Accuracy with Dynamic Chunking and Optimized Vector Search. *Orclever Proceedings of Research and Development*, *5*(1), 215–225. https://doi.org/10.56038/oprd.v5i1.516

Vasilios Mavroudis. (2024). *LangChain.* Alan Turing Institute. https://www.researchgate.net/publication/385681151_LangChain

Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., & Wei, F. (2022). *Text Embeddings by Weakly-Supervised Contrastive Pre-training.* https://doi.org/10.48550/arXiv.2212.03533

Wu, Y [Yaxiong], Liang, S., Zhang, C., Wang, Y., Zhang, Y., Guo, H., Tang, R., & Liu, Y. (2025). *From Human Memory to AI Memory: A Survey on Memory Mechanisms in the Era of LLMs.* https://doi.org/10.48550/arXiv.2504.15965

Yuan, Y., Liu, C., Yuan, J., Sun, G., Li, S., & Zhang, M. (2024). *A Hybrid RAG System with Comprehensive Enhancement on Complex Reasoning.* https://doi.org/10.48550/arXiv.2408.05141

10 Appendices