Retrieval Augmented Generation for QA Tasks

Teiboklang Chyne
234156019
c.teiboklang@iitg.ac.in

Vaibhav Dewangan
234156020
v.dewangan@iitg.ac.in

Neeraj Kumar
234156014
neerajks@iitg.ac.in

Abstract

Retrieval-Augmented Generation (RAG) has emerged as a prominent paradigm for enhancing Question Answering (QA) tasks. RAG's superiority lies in its ability to augment the knowledge stored within Language Model parameters by retrieving external information. This approach surpasses the limitations of relying solely on internal knowledge, thereby outperforming task-specific models. Despite its effectiveness, RAG systems are intricate and offer numerous avenues for enhancement. To optimize RAG models, we delve into the impact of chunk size, embeddings, and re-ranking strategies. Through extensive QA task experimentation on Language Models, we aim to evaluate RAG performance while varying these parameters. Our objective is to discern the parameters' influence and uncover strategies for bolstering RAG's QA capabilities.

1 Introduction

Large Language Models (LLMs) have revolutionized the way people seek information online, from searching to directly asking chatbots for answers. Although recent studies have shown their state-of-the-art capabilities in question answering (QA) in general and medical domains, the training corpora of LLMs might not include the latest knowledge.

By providing LLMs with relevant documents retrieved from up-to-date and trustworthy collections, Retrieval-Augmented Generation (RAG) has the potential to address the above challenges. As such, RAG has already been quickly implemented in various QA systems. However, a complete RAG system contains several flexible modules in the retrieval and vectorization, such as chunk size, embeddings, and re-rankings, but the best practices for tuning these components are still unclear, hindering their optimal adoption in QA.

To find out the best combination of these to improve the RAG system and to systematically evaluate how different components in RAG affect its performance, we first build a synthetic dataset of questions and associated contexts. The method is to get elements from our knowledge base and ask an LLM to generate questions based on these documents. Next, we set up other LLM agents to act as quality filters for the generated QA couples: each of them will act as the filter for a specific flaw. We use Mixtral for QA couple generation because it has excellent performance in leaderboards such as Chatbot Arena. We systematically score functions with all these agents, and whenever the score is too low for any one of the agents, we eliminate the question from our evaluation dataset.

In the preprocessing step, we split the documents from our knowledge base into smaller chunks, ensuring they are semantically relevant snippets. We then use the Retriever, which acts like an internal search engine, to return the most relevant documents from our knowledge base based on the user query.

For the knowledge base, we use Langchain vector databases due to its convenient FAISS index and ability to retain document metadata throughout processing. In the Reader component, the LLM reads the retrieved documents to formulate its answer, and options for improvement include switching reranking on or off, adjusting the size of chunks, and changing the embedding model.

To benchmark the RAG system, we evaluate its output on the evaluation dataset using a judge agent focusing on faithfulness, which is the best end-to-end metric of system performance. We use GPT4 as a judge. Prompting the LLM to output rationale before giving its final score helps formalize and elaborate its judgment.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) was proposed by Lewis et al. (2020) to enhance the gen-

question	answer	groundedness_score	relevance_score	standalone_score
Who started the business called Kinko's?\n	Paul Orfalea started the business called Kinko's with a $5,000 loan co-signed by his father in 1969. He initially rented a small garage and sold about $2,000 worth of services daily with the help of a few friends.	5.0	5.0	5.0
What is an example of a future goal that young children may have?\n	An example of a future goal that young children may have is driving a tractor when they grow up. This goal reflects the interests and values of a young child, who may not yet fully understand the complexities and challenges of pursuing such a goal in the context of their future lives.	3.0	2.0	5.0
What percentage of American households have a money market deposit account?\n	According to the context, approximately 15 percent of American households have a money market deposit account.	5.0	4.0	5.0

Figure 1: Example Questions With Scores

eration performance on knowledge-intensive tasks, mitigating the problem of hallucinations as LLMs are grounded in given contexts. Additionally, RAG can provide up-to-date knowledge that might not be encoded by the LLMs. Several follow-up studies have been conducted to improve upon the vanilla RAG (Borgeaud et al., 2022; Ram et al., 2023; Gao et al., 2023; Jiang et al., 2023; Mialon et al., 2023).

2.2 RAG Benchmarking

"ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems" by Khattab et al. (2021), introduces ARES (Automated RAG Evaluation System), a solution that automates the evaluation process to address the limitations of manual annotation. ARES evaluates various dimensions including Context Relevance, assessing how well the retrieved context aligns with the user query; Answer Faithfulness, determining if the generated answer is faithful to the retrieved information; and Answer Relevance, evaluating the relevance of the generated answer to the query. ARES achieves this by creating synthetic training data and fine-tuning lightweight LM judges to evaluate individual RAG components.

2.3 QA Evaluation

"Natural Questions: A Benchmark for Question Answering Research" by Kwiatkowski et al. (2019) annotation methodology, evaluation metrics, and role as a benchmark significantly improved QA research, fostering advancements in handling open-domain questions and enhancing system performance.

3 Datasets

For our dataset, we utilized six books focusing on wealth, wisdom, and enlightenment: "Learn to Earn" by Peter Lynch and John Rothchild, "The Education of a Value Investor" by Guy Spier, "Rich Dad Poor Dad" by Robert T. Kiyosaki, "The Psychology of Money" by Morgan Housel, "The Total Money Makeover" by Dave Ramsey, and "The Millionaire Next Door" by Thomas J. Stanley.

To prepare the input data, we transformed these books into CSV files containing text and metadata. Each CSV file represents a single book and includes two columns: one for the text content and another for metadata such as the source document number. This format enables structured access to the book content for further processing and analysis.

4 Experimental Steps

4.1 Load Knowledge Base

We read each CSV file, which contains the raw text data, and convert this text into LangchainDocument objects. These objects represent individual documents in our corpus. Next, we address the need for granularity by splitting the LangchainDocument objects into smaller chunks. To achieve this, we employ Langchain's RecursiveCharacterTextSplitter, which allows us to segment the documents based on different separators such as newlines, periods, and spaces. The resulting processed documents, organized into these smaller chunks, are stored in a list. This list serves as our knowledge base.

4.2 Generate Questions

We utilize "mistralai/Mixtral-8x7B-Instruct-v0.1" from Hugging Face to create context-based question-answer pairs from the dataset by randomly sampling contexts. The number of questions generated can be adjusted, with our current setup producing 100 questions per iteration. Extracting the questions and answers from the model's output, we ensure that the answers are of appropriate length. These QA pairs are then stored in a list for subsequent steps.

4.3 Filter Questions

The questions produced may have various issues, so it's essential to conduct a quality check before confirming them. To achieve this, we use the Mix-

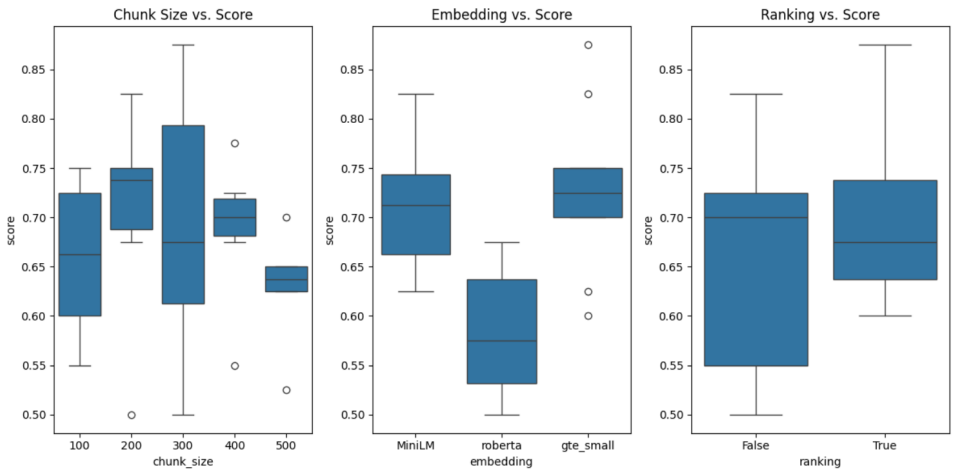

Figure 2: Relations between the parameters and the final score

tralAI to assess each question based on specific criteria:

- **Groundedness**: can the question be answered from the given context?
- **Relevance**: is the question relevant to users?
- **Stand-alone**: is the question understandable free of any context, for someone with domain knowledge/Internet access?

If a question receives a low score in any of these criteria, we exclude it from our evaluation dataset.

4.4 Setup the RAG Model

We configure the desired LLM model for processing question answers, opting for the "HuggingFaceH4/zephyr-7b-beta" model. This model is employed to generate answers based on the previously generated questions.

4.5 Evaluation

We try different combinations of chunk size, embeddings, and rerank settings for the RAG model. The chunk sizes used are 100, 200, 300, 400, and 500. The embeddings used are "thenlper/gte-small","sentence-transformers/all-MiniLM-L6-v2", and "sentence-transformers/roberta-base-nli-stsb-mean-tokens" available in HuggingFace. The re-ranker used is "colbert-ir/colbertv2.0", again from HuggingFace. For each combination of settings, we load the knowledge base using the specified chunk size and embedding model. We then run the RAG model with the specified settings, including the use of a reranker model if rerank is set to True. After running the RAG model, we evaluate the generated answers using "gpt-4-turbo-preview" from OpenAI based only on faithfulness, which is the best end-to-end metric of our system's performance, by providing the original answer and source documents as a reference.

5 Results

5.1 Descriptive Statistics

We calculate the mean, median, and standard deviation of the scores for each parameter type.

Chunk Size	Mean	Median	Std Deviation
100	0.65833	0.6625	0.084656
200	0.704167	0.7375	0.111149
300	0.691667	0.6750	0.140238
400	0.687500	0.7000	0.075416
500	0.629167	0.6375	0.057915

Table 1: Summary Statistics for Different Chunk Sizes

The mean and median scores for language model faithfulness are highest when the chunk size is 200. This suggests that a chunk size of 200 provides a good balance between context length and computational efficiency. Therefore, we could further explore chunk sizes around this size, as we may obtain better results.

As the chunk size increases beyond 200, both the mean and median scores start to decline. This could be due to larger chunks introducing noise or irrelevant context.

We can observe that the standard deviation (std) increases as the chunk size grows. Larger chunks may capture more diverse content, resulting in

greater variability in scores.

Embedding	Mean	Median	Std Deviation
MiniLM	0.7125	0.7125	0.061520
roberta	0.5825	0.5750	0.067752
gte-small	0.7275	0.7250	0.082031

Table 2: Summary Statistics for Different Sentence Transformers

GTE-Small demonstrates superior performance compared to MiniLM and RoBERTa. However, it also exhibits the highest standard deviation, indicating greater variability in performance across different contexts.

Re-Ranking	Mean	Median	Std Dtn
False	0.651667	0.700	0.103711
True	0.696667	0.675	0.083381

Table 3: Summary Statistics for Re-ranking

When re-ranking is set to "True," the mean score increases from 0.651667 to 0.696667, indicating a potential improvement in overall performance. However, the median score decreases from 0.700 to 0.675, suggesting that while re-ranking may enhance the mean score, it could also introduce more variability in individual scores. This variability is reflected in the standard deviation (std), which is significantly lower when re-ranking is "True" (0.038381) compared to when it is "False" (0.103711). The lower standard deviation indicates less variability in scores when re-ranking is applied, highlighting a trade-off between potentially higher mean scores and increased variability in results.

5.2 One-way ANOVA test

Parameter	F-Statistic	P-Value
Chunk Size	0.56911	0.68741
Embedding	12.62896	0.00013
Re-ranking	1.71529	0.20094

Table 4: One-way ANOVA test results

From the results of a one-way ANOVA test for three different parameters—Chunk Size, Embedding, and Re-ranking—we can draw several conclusions. The parameter that exerts the most significant impact on the outcome is the embedding type. This inference is supported by the low p-value (0.00013) associated with the embedding parameter, indicating its likely significant effect on the outcome. The F-Statistic, which measures the variability between group means relative to the variability within groups, shows the highest value for the embedding parameter (12.62896), further confirming its impact. While the re-ranking parameter has a higher p-value (0.20094), suggesting it may not be as significant in affecting the outcome compared to embedding, it still has a higher impact than chunk size. Overall, the results of the one-way ANOVA test indicate that the choice of embedding is statistically significant, while the choice of chunk size and re-ranking is statistically insignificant.

6 Limitations

6.1 Increased Complexity

Collaboratively using multiple LLMs increases the complexity of the problem-solving process. Coordinating their outputs, managing conflicting responses, and integrating their contributions is challenging.

6.2 Dependency on Prompt Quality

The quality of the prompts provided to the LLMs significantly influence the output. Poorly crafted prompts leads to misinterpretations by the agents.

6.3 Scalability and Resource Intensity

Running multiple LLMs requires significant computational resources, including processing power and memory. Scaling the project, increasing its corpora and increasing the number of questions to evaluate, may require significant computational resources and infrastructure. This can be a limitation for individuals with limited resources or time constraints.

6.4 Commercial LLM

The language models (LLMs) utilized in this study are proprietary, precluding substantial modifications compared to open-source LLMs. Additionally, the embeddings employed are commercially licensed, which could potentially present a constraint or limitation in our research methodology.

7 GitHub Repository of Project

Retrieval Augmented Generation for QA Tasks

8 References

- Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

- ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems
- Natural Questions: A Benchmark for Question Answering Research
- Colab Notebook: RAG Evaluation