An Efficient Approach for Answering Not Readily Attainable Questions for RAG-based Applications

Zhengdao Chen[1] and Christian Heumann[1] and Matthias Aßenmacher[1,2]

[1]Department of Statistics, LMU Munich,
[2]Munich Center for Machine Learning (MCML), LMU Munich
Correspondence: matthias@stat.uni-muenchen.de

Abstract

Retrieval-augmented generation (RAG) is an established method for addressing challenges in applying large language models (LLMs), such as ensuring timeliness, incorporating domain-specific expertise, and minimizing hallucinations. However, the effective application of data-augmented LLMs remains challenging due to, e.g., reliance on retriever performance, token-limit restrictions for the input, or the inherent difficulty of global questions directed at large text corpora. Despite various efforts to address these challenges, there are still instances where finding correct answers to certain questions remains elusive. Moreover, as more modules are added to the RAG pipeline, its complexity and latency increase, so that the achieved performance improvements may become less practically significant. Based on these observations, we propose an efficient approach to addressing the issue of not readily attainable questions in a pragmatic way: by collecting questions with incorrectly generated answers, preparing the correct answers offline, and prepending a module for semantic search among the prepared question-answer pairs to the RAG system. If we consider a traditional RAG system an open-book exam, this QA search module can be likened to an open-question exam, similar to a driver's license test.

1 Introduction

Retrieval-augmented generation (RAG) is widely utilized for Large language models (LLMs) to address associated challenges, such as timeliness and lack of awareness of domain-specific knowledge, particularly when it comes to private or on-premise information, as this can impact their reliability, leading to issues with faithfulness and the generation of hallucinations. The main idea of RAG is to condition a pre-trained LLM on relevant information retrieved from external data sources based on a user's input during the generation process (Lewis et al., 2021; Ram et al., 2023).

However, RAG systems encounter their own set of challenges. Firstly, they inherit certain limitations from LLMs, such as constraints on input tokens and difficulties in effectively addressing broad questions within extensive text corpora. Secondly, a RAG system is inherently complex, comprising numerous components that require significant time and expertise to configure and integrate seamlessly. Thirdly, the retrieval process can introduce latency, impacting response times. Additionally, training both the retriever and generation models demands meticulous tuning and substantial computational resources. Furthermore, variations in the underlying data can complicate integration and scalability efforts. Approaches addressing these challenges include optimizing the storage and semantic representation of external data, aligning queries and documents to accurately retrieve the relevant external information (Wang et al., 2023; Ma et al., 2023a; Zheng et al., 2024), improving imperfect retrieval performance by analyzing, identifying and removing irrelevant information (Wang et al., 2025), generating query samples to train retrievers for specific domains (Dai et al., 2022), fine-tuning retriever (Shi et al., 2023), compressing (Yang et al., 2023; Xu et al., 2023; Ma et al., 2023b), re-ranking retrieved information before passing it to LLMs (Zhuang et al., 2023), and fine-tuning LLMs for generating results (Cheng et al., 2023; Kang et al., 2023; Li et al., 2023b). While each of these techniques, whether employed individually or in combination, has the potential to enhance performance, certain questions persist for which generating an accurate response remains challenging.

We propose the integration of an additional question-answer pairs (QA-pairs)-based search module, designed to be incorporated as a preparatory component preceding the RAG system (cf. Figure 1), in which a pre-prepared list of existing QA-pairs is directly utilized as context by LLMs to

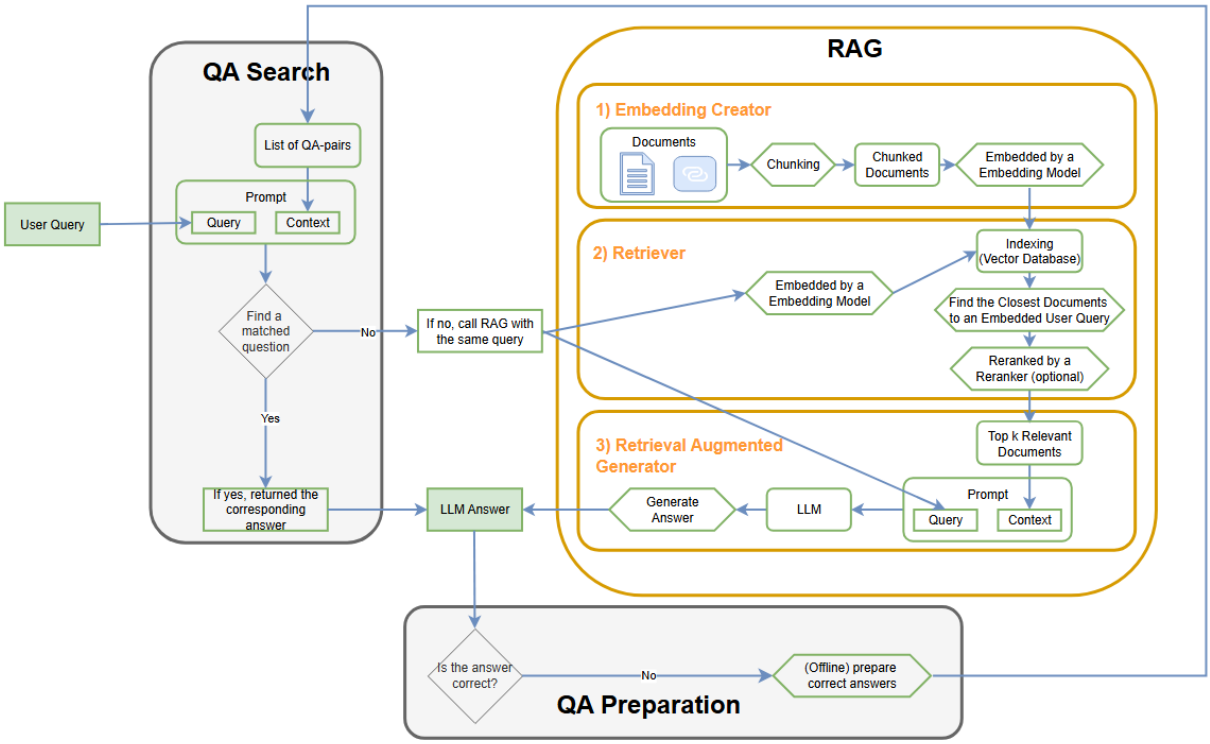

Figure 1: RAG with Integration of a QA Search Module

address user queries. If a question with the same semantic meaning is identified within the list, the associated answer is returned directly. Conversely, if no matching question is found, the user query is forwarded to the standard RAG process. The list of QA-pairs can be created either a priori or continuously during the process. As illustrated in Figure 1, questions for which the RAG system fails to generate correct answers can be collected. Subsequently, the expected answers to these questions can be prepared offline and appended to the list within the QA search module.

Contributions. Our proposed QA search module offers three key contributions:

1. It significantly enhances the overall accuracy of the RAG system. Through experiments conducted with various LLMs and QA datasets, we demonstrate the effectiveness of semantic search. For questions that are incorrectly answered by RAG, we curate correct responses and compile a QA-pairs catalog, enabling precise and efficient semantic retrieval.

2. The QA search module reduces average processing time and resource consumption, as evidenced by our experimental results, which highlight its efficiency.

3. All these benefits are achieved through a lightweight extension of the RAG system, ensuring that the module remains easy to update and scale as needed.

2 Background and Related Work

To tackle challenges such as timeliness and the lack of awareness regarding private, domain-specific knowledge in pre-trained LLMs, various techniques have been developed. Among these, RAG is gaining significant attention and widespread application. While the implementation of RAG can vary in practice, the fundamental concept remains consistent: for a given input request, relevant documents are retrieved from external knowledge sources, incorporated with the input to assist LLMs during the generation process.

REALM (Guu et al., 2020) employs a masked language model (MLM) and fine-tunes it for open domain question-answering (Open-QA). Similarly focused on Open-QA, RAG (Lewis et al., 2021) trains an autoregressive language model (ALM), while Atlas (Izacard et al., 2022) utilizes an encoder-decoder LLM. In-Context RALM (Ram et al., 2023) and Replug (Shi et al., 2023) simplify the process further by treating the LLM as a black box, eliminating the need for additional training. In these approaches, retrieved information is prepended as context to the query before

being passed to the off-the-shelf LLMs for generation. kNN-LM (Khandelwal et al., 2020) offers an even more practical approach by treating a question as a prefix to its answer. It transforms the task of finding the correct answer into locating the most similar expressions to the input question within an embedding space. This space is created by a LLM on a knowledge dataset. The k most similar results are then interpolated with the original model outputs to construct the final answer. Moreover, both Zero-Shot-Learners (Wei et al., 2022) and Zero-Shot-Reasoners (Kojima et al., 2023) demonstrate exceptional language understanding capabilities inherent in LLMs, particularly in the zero-shot implementation of the so-called Chain of Thought (CoT, Wei et al., 2023).

3 Approach

To answer a question x using the classic RAG paradigm, two steps are involved, retrieval and generating. First, given a question x, the most relevant documents z are retrieved from a knowledge corpus $\mathcal{Z}$. This step can be represented by a probability distribution $p(z \mid x), z \in \mathcal{Z}$. Next, both the input question x and the retrieved documents z are provided to a LLM to generate the answer y. This generation step can be modeled as $p(y \mid z, x)$. By considering the retrieved documents z as a latent variable and marginalizing over all documents in $\mathcal{Z}$, the entire process can be formalized as:

$$p(y \mid x) = \sum_{z \in \mathcal{Z}} p(y \mid z, x)\, p(z \mid x) \qquad (1)$$

Our new approach takes the simplification one step further by providing an existing list of QA-pairs as context, instead of retrieving most similar external documents to each given question and providing these as context to the question to a LLM to generate the answer. By doing so, we leverage the model to identify the most closely matched questions from the list and return the corresponding answer to new input questions. This approach transforms a QA task into a task of finding semantically similar expressions. It's common for the input question to differ in wording from those in the existing QA-list. However, LLMs excel at understanding and interpreting the semantic meaning of text, making them well-suited for this task.

We begin with an existing list of QA-pairs $\mathcal{Z} = \bigcup_{i=1}^{n} \{(x_i, y_i)\}$, which is assumed to encompass the most frequently asked questions. Instead of employing a retrieval step, we provide this entire set as context to LLMs. This implies that $p(z \mid x) = 1$, resulting in

$$p(y \mid x) = \sum_{z \in \mathcal{Z}} p(y \mid z, x) \qquad (2)$$

The initial list does not need to be exhaustive. If we encounter a new question x', that is not already included in the set of questions ($x' \notin \bigcup_{(x_i, y_i) \in \mathcal{Z}} \{x_i\}$), we can prepare the corresponding answer y' and update the the QA-list to $\mathcal{Z}' = \mathcal{Z} \cup \{(x', y')\}$.

When the QA-list is too extensive to fit entirely as context into LLMs due to input token limit constraints, we can assign a reference number to each QA-pair. This forms a expanded set $\mathcal{Z}_{exp} = \bigcup_{i=1}^{n} \{(x_i, y_i, n_i)\}$. Instead of passing the list of QA-pairs to the LLMs, we provide a list of question-number pairs (QN-pairs) $\mathcal{Z}_{red} = \bigcup_{i=1}^{n} \{(x_i, n_i)\}$, which reduces the context size. The LLMs then return the corresponding reference number if the input question matches one in the list; otherwise, they return a special number, such as "-1". Using the returned reference number, we can identify the actual answer by performing a lookup in the expanded QA-list $\mathcal{Z}_{exp}$: $y_i = f_{lkp}(\mathcal{Z}_{exp}, n_i)$.

If the list of the QN-pairs still exceeds the input token limit even after replacing actual answers with reference numbers, we can divide the list into small blocks, each within the token limit. We can then process these blocks sequentially using LLMs. The loop can be terminated early if a valid reference number is returned.

The remaining question is how effectively LLMs can interpret various expressions of an input question x, which may differ in length and form. Specifically, we aim to assess their ability to identify the expression with the same meaning in set $\mathcal{Z}_{exp}$ and return its corresponding reference number. To achieve this, we designed the question-question similarity check in a lightweight manner by implementing a direct search within a prompt, rather than relying on the cosine similarity of embedded questions. This approach leverages the strength of LLMs in capturing semantic information from diverse expressions effectively.

4 Feasibility Assessment Experiments

We assessed the feasibility of our approach by designing experiments across three dimensions: (i) Open-QA datasets, (ii) LLMs for rephrasing original questions and (iii) LLMs for answering original and rephrased questions.

4.1 Experimental Setup

Datasets We selected four publicly available QA datasets for our study. Two of these are in English: OpenBookQA (Mihaylov et al., 2018), CMU-Wiki-QA (Smith et al., 2008), one in Chinese: CMMLU (Li et al., 2023a) and one in German: LHM-Dienstleistungen-QA (Schröder et al., 2022).

LLMs The key success factor of our approach is the ability of LLMs to effectively interpret various expressions of a given question and identify it within a context. To evaluate this we selected several LLMs to rephrase the original questions and to answer both the original and rephrased questions.

These models are GPT3.5, GPT4o (OpenAI, 2023) and seven open-source models: Llama 3.1:8B, Llama 3.3:70B (META, 2024), Qwen2:7B, Qwen2.5:7B (Group, 2024), Mixtral8:7B, Mixtral8:22B (Jiang et al., 2024), Gemma2:9B, Gemma2:27B (DeepMind, 2024).

4.2 Implementation Details

We divided the entire implementation process into five distinct steps.

Preparing Datasets The four QA datasets were sourced from Hugging Face. We selected a random sample of 150 QA-pairs from each dataset and assigned a reference number to each question.

Rephrasing Questions We utilized nine selected LLMs to generate six different variants of each original question across all datasets. These variants maintain the same meaning as the original question. Specifically, three variants were generated without any length restrictions, while the other three were limited to a maximum of 30 characters. The shorter variants were designed to mimic situations where users prefer to keep questions concise. For each original question, we obtained a total of 55 different expressions, including the original question itself: 1 (original question) + [3 (rephrased variants) + 3 (rephrased short variants)] * 9 (models for rephrasing) = 55. More details see Table 2. The prompt template utilized is available in Figure 6.

Algorithm 1 Answering Questions by QA Search

$N_{qn_blocks} \leftarrow ceil(n_{total_qn}/block_size)$
for $each\ question\ q$ **do**
  for $i\ from\ 0\ to\ N_{qn_blocks}$ **do**
    $C_i \leftarrow create_context(qn_block_i)$
    $a_n \leftarrow call_qa_search(q, C_i)$
    if $a_n > -1$ **then**
      $answer \leftarrow f_{lkp}(L_{QAN}, a_n)$
      $return\ answer$
    end if
  end for
end for

Answering Questions We used ten LLMs to answer all question variants, by utilizing the list of original QN-pairs as context. As Algorithm 1 shows, if a match is not found in the list, the corresponding reference number is returned, otherwise "-1" is returned. The prompt template utilized is available in Figure 7.

Both Mixtral8 models returned numbers with wrapping text, such as "The reference number of its corresponding answer is: 1037", instead of "1037". An additional script was created to remove the extraneous text.

Using all 150 original QN-pairs as context worked well with both GPT models. However, for the open-source models, after several rounds of experimentation, we decided to split the full list into five blocks with 30 questions each. We then looped over these blocks until a reference number is returned. Due to text wrapping issues, we needed to collect results from all five blocks without breaking the loop early when using both Mixtral8 models.

Evaluating Answers We compared the reference numbers returned by the LLMs with the ground truth. If they match, we assigned a score of 1; otherwise, we assigned a score of 0. We then aggregated these scores to calculate the accuracy rate of the answers across all question variants for each model.

Robustness Testing with Different List and Block Sizes To demonstrate the robustness of performance as well as resource and time consumption, we conducted additional experiments with varying total numbers of QA-pairs (300, 600, 900, 1200, 1500) and different numbers of QA-pairs (30, 50, 100, 150, 300) used as query context.

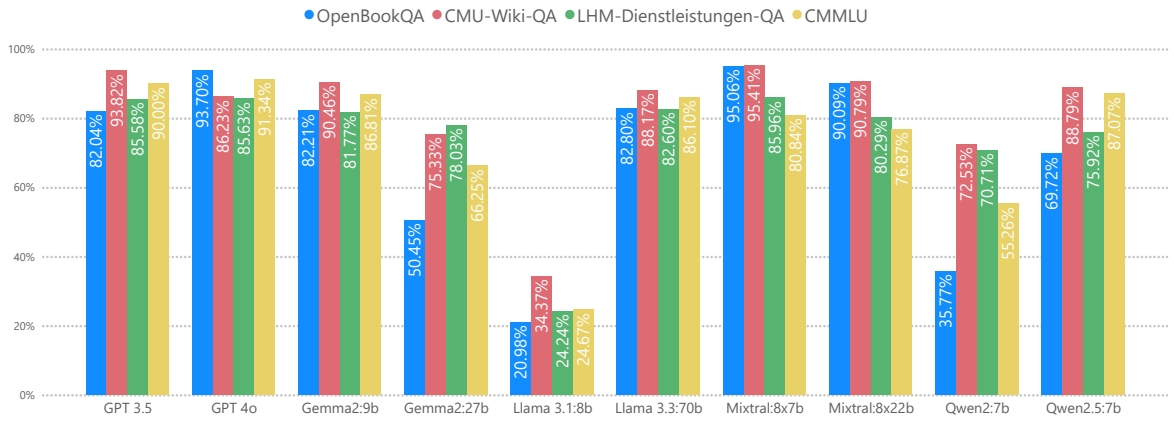

Figure 2: Average Percentage of Correctly Answered Questions for Four QA Datasets by Each Model

4.3 Preliminary Experimental Results

Comprehensive results for the feasibility experiment for the four QA datasets are available in Tables 3-6. Additionally, we calculated the average answering accuracy of each applied model across all four dataset. As illustrated in Figure 2, all models demonstrate consistent performance across the four datasets, which include different languages: English, Chinese, German. This consistency holds true regardless of whether the models achieve high or low accuracy. The GPT models, in particular, exhibit best overall performance. This is not only due to their average accuracy rates exceeding 80%, but also because they do not require the question list to be divided into blocks for iterative processing.

Among the open-source models, Mixtral8:7B, Gemma2:9B and Llama 3.3:70B achieve an accuracy rate greater than 80% across all datasets. Mixtral8:22B follows closely, with only one dataset falling below 80%, while Qwen2.5:7B has two datasets under this threshold. In contrast, the average accuracy rates for Gemma2:27B and Qwen2:7B fall below 80%, ranging between 36% and 78%. Surprisingly, Llama 3.1:8B performs the worst, with all accuracy rates below 35%.

By expanding the total number of QA-pairs in the catalog from 300 to 1500, while maintaining a fixed context size of 150 QAs, and using GPT4o as the answering LLM, we observed, as expected, no significant increase in CPU, GPU, or memory usage. However, this expansion resulted in longer average response time, increasing from 1.39 seconds per query to 6.15 seconds per query, due to increased number of max. needed requests. As illustrated in Figure 3, the mean accuracy experienced a slight decline as the catalog size increased, dropping from 93.65% to 89.35%. This reduction in performance can be attributed to the higher likelihood of encountering questions with similar but not identical meanings as the catalog grows, which increases the difficulty of semantic search for the LLM.

If, instead, we fixed the total number of QA-pairs in the question catalog but expanded the context size from 30 to 300, we observed, as shown in Figure 4, again, no significant increase in CPU, GPU, or memory usage. And as expected, the average response time decreased from 4.36 seconds to 1.28 seconds per query. The mean accuracy improved significantly, increasing from 88.13% to 95.48%. This improvement can be attributed to the fact that, as long as the expanded context size remains within the token limit of the LLM, having more questions with similar but not identical meanings in a single context allows the LLM to better identify and select the one with the closest semantic match.

The detailed metrics for both Figure 3 and 4 are provided in Table 7 and 8. The four QA datasets, all codes and generated outputs for this experiment are available on GitHub[1].

5 Comparative Experiments between RAG and RAG with a QA Search Module

After successfully demonstrating the feasibility of a QA search, we conducted another group of experiments to showcase the enhanced effectiveness

[1] https://github.com/chenzhengdao/pattern-of-driving-license-written-exam, https://github.com/chenzhengdao/pattern-of-driving-license-written-exam-compare-with-rag

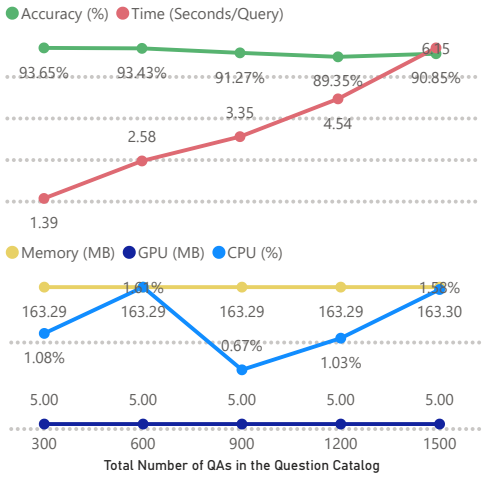

Figure 3: Accuracy, Resource and Time Consumption Across Varying Total Numbers of Questions in the Question Catalog, with 150 Questions as QA Search Context and GPT4o as the Answering LLM

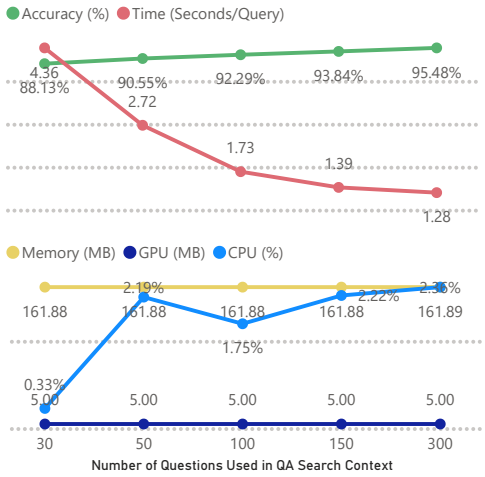

Figure 4: Accuracy, Resource and Time Consumption Across Varying Number of Questions Used as QA Search Context, with 300 as Total Number of Questions in Question Catalog and GPT4o as the Answering LLM

and efficiency of a RAG system by integrating a QA search module. To achieve this, we setup experiments for the following four approaches and compared their results: i) RAG: baseline RAG, ii) QA search: as evaluated in the previous section, iii) RAG + QA search: applying QA search to questions incorrectly answered by RAG iv) QA search + RAG: applying RAG to questions incorrectly answered by QA search. The last approach is exactly what Figure 1 illustrates.

5.1 Experimental Setup

RAG As illustrated in the top-right section of Figure 1, the RAG system used in the following experiments consists of three main modules, inspired by (Lewis et al., 2021) and (Gao et al., 2024): (i) an embedding creator, (ii) a retriever, and (iii) a retrieval-augmented generator. Initially, documents are segmented into smaller chunks, which are then transformed into embeddings using an embedding model. These embeddings are stored in a vector database as a knowledge base through an indexing algorithm, making them ready for efficient retrieval. When a user submits a query to the RAG system, the query is embedded using the same embedding model and sent to the vector database to identify the top k most relevant documents. In some cases, a reranker, typically another LLM, may be employed to reorder the retrieved documents for improved relevance. Finally, a prompt is constructed by combining the user query with the top k documents as context, which is then sent to an LLM to generate the final response to the user's query.

Datasets The four QA datasets used for the feasibility assessment in last section are unsuitable for this experiment, as they consist solely of QA-pairs and do not include the corresponding text blocks from which the answers can be generated. We chose two other publicly available document sets: (i) it-tech-doc, an IT-technical document set and (ii) financial-news, a financial news document set.

Building Synthetic QA-pairs We devided each document from both document sets into chunks, with a size of 2000 tokens and an overlap of 200 tokens. For each document chunk, we utilized two LLMs (GPT3.5 and GPT4o) to generate synthetic QA-pairs. We then evaluated the quality of the generated QA-pairs, assigning a score between 1 (poor) and 5 (excellent) based on three criteria: (i) **Groundedness**: can the question be answered from the given context? (ii) **Relevance**: is the question relevant for a given topic? (iii) **Stand-alone:** is the question understandable free of any context? We retained QA-pairs with best scores across all three criteria and selected 150 samples from each QA set for further processing. The prompt templates utilized here are available in Figure 8-11.

Building Test Cases for Evaluating RAG Evaluating a RAG system can be complex due to various factors that influence its overall performance. But

it is not the primary focus of this experiment to try out different combinations of influence factors. For this work we simply used two chunk sizes (200 and 400) to create chunking, applied one embedding model (GTE-small, a variant from the General Text Embedding (GTE) series developed by Alibaba Group (Li et al., 2023c)) to embed and retrieve embedded chunks. We then applied two approaches to the retrieved chunks: one without a reranker and one with a reranker (colbertv2.0, a model developed at Stanford University (Santhanam et al., 2022)). Furthermore, we used two LLMs (the two top performers from the feasibility experiment in Section 4: GPT4o and Llama 3.3:70B) to generate the answer to a given question based on the found chunks and another LLM for evaluating the generated answers. This in total made up 16 test cases for the baseline RAG evaluation: 2 (LLMs for QA generation) $\times$ 1 (LLM for QA evaluation) $\times$ 2 (chunk sizes for embedding) $\times$ 1 (embedding model) $\times$ 2 (options for reranking, with or without) $\times$ 2 (LLMs for answer generation) = 16. Finally, for each of the 16 test cases we obtained an average accuracy score for generated answers across all 150 sample questions, and made the average of the 16 average accuracy scores as the final accuracy to the baseline RAG. Details about the 16 combinations for both document sets can be found in Table 9- 10.

Building Test Cases for Evaluating QA Search After the comprehensive feasibility experiments in Section 4, we intended not to construct too many test cases for the QA search testing. To each of the synthetic QA-pairs sample sets, we applied one LLM (GPT4o) for rephrasing the original question, and two LLMs (GPT4o and Llama 3.3:70B) for QA search. This made in total 8 test cases: 2 (LLMs for synthetic QA generation) $\times$ 1 (LLM for synthetic QA evaluation) $\times$ 1 (LLM for rephrasing question) $\times$ 2 (question types, original and rephrased) $\times$ 2 (LLMs for QA search) = 8. Details about the 8 test cases for both document sets can be found in Table 11- 12.

5.2 Implementation Details

For each of the document sets (it-tech-doc and financial-news), we applied the following four approaches:

Answering Questions with RAG For each of the 16 test cases, we answered the sampled questions by following the RAG pipeline described in Figure 1 and detailed in Algorithm 2. The prompt templates used in this process are provided in Figure 12.

Unlike the binary evaluation used for QA search, as described in Subsection 4.2, the answers generated by RAG are evaluated using a scoring scale from 1 (worst) to 5 (best). Answers receiving a score of 4 or 5 are considered correct, while those scoring below 4 are deemed incorrect. The prompt template used for this evaluation is provided in Figure 13.

Algorithm 2 Answering Questions by RAG

```
for each question q do
    q_e ← embed(q)
    D_k ← retrieve(q_e, D, k)
    if reranking then D_kr ← rerank(D_k)
        a ← generate(D_kr, q)
    else
        a ← generate(D_k, q)
    end if
    Save answer a
end for
```

Applying QA Search to Questions Incorrectly Answered by RAG In the next step, we collected all the questions that were incorrectly answered in the previous phase, rephrased them, and then applied Algorithm 1 to the rephrased questions, using the list of original QA-pairs as context.

Answering Questions with QA Search For all 8 test cases of QA search, we applied the semantic search of questions among the list of original QA-pairs as described in Section 3-4.

Applying RAG to Questions Incorrectly Answered by QA Search We gathered all the questions that were incorrectly answered by the QA search in last step. We then applied Algorithm 2 to each of them to generate answer, and calculated the final accuracy.

The two document sets, all codes and generated outputs for this experiment are available at this Git repository 2.

5.3 Main Experimental Results

5.3.1 Accuracy Improvement

Based on Figure 5, it is evident that, without any parameter tuning on either side, the three alternative approaches consistently outperform the baseline RAG across both document sets. By applying the

Measure	GPU (MB)		Memory (MB)		CPU (%)		Time (Sec/query)	
Method	RAG	QA Search	RAG	QA Search	RAG	QA Search	RAG	QA Search
it-tech-doc	899	6	1,741	161	5.92%	2.09%	11.68	0.81
financial-news	899	6	1,865	162	5.82%	2.00%	10.93	0.73

Table 1: Mean Resource & Time Consumption for RAG and QA Search

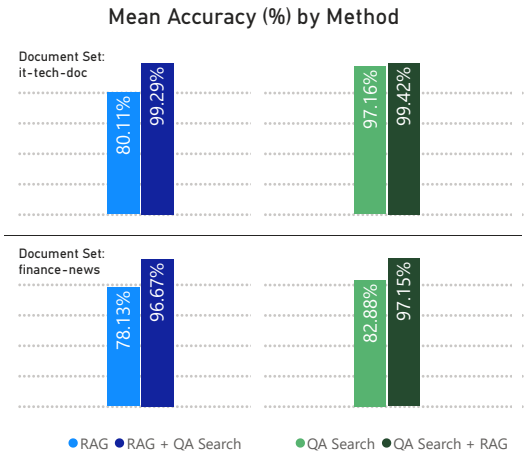

Figure 5: Mean Accuracy of Four Methods: i) RAG, ii) RAG + QA Search, iii) QA Search, iv) QA Search + RAG

QA search method to the questions incorrectly answered by the baseline RAG, the accuracy can be significantly improved—from 80.11% to 99.29% for one document set and from 78.13% to 96.67% for the other.

Remarkably, the QA search approach alone delivers superior results compared to the baseline RAG, with accuracy improvements of 17% and 4%, respectively. Furthermore, the final combined approach (QA search + RAG), achieves the highest mean accuracy, reaching 99.42% for it-tech-doc and 97.15% for financial-news. This optimal approach is precisely the one depicted in Figure 1.

The detailed average accuracy for all test cases is provided in Tables 9–12.

5.3.2 Efficiency Improvement

In terms of computational resource consumption per user query, the QA search approach demonstrates significant advantages over the traditional RAG. As shown in Table 1, querying with QA search requires no GPU resources (the 6 MB listed in the table is attributed solely to operating system overhead), whereas the RAG setup described in Subsection 5.1 demands nearly 900 MB of GPU memory. Additionally, RAG exhibits over 10 times higher memory consumption and nearly 3 times greater CPU usage compared to QA search.

Thanks to its lightweight architecture, the QA search approach achieves a significantly shorter average response time compared to RAG —under 1 second (0.81 or 0.73 seconds) versus over 10 seconds (11.68 or 10.93 seconds) for RAG. It is important to note that the reported response times are based on a question catalog containing 150 entries, all consolidated into a single query context. For a detailed comparison of response times across different catalog sizes or context configurations, please refer to Figures 3 and 4.

Detailed average resource consumption metrics are provided in Tables 13–16. It is worth noting that in our experiments, local open-source LLMs used for answering questions are hosted as separate standalone processes and are not part of the query processing pipeline. Consequently, their GPU consumption is excluded from the resource measurements for both RAG and QA search.

5.3.3 Expert Answer instead of Hallucination

For QA-pairs used in the QA search module, answers can be thoroughly prepared in advance. For instance, we can engage experts to formulate or review these answers. Additionally, we can enhance the responses by including supplementary information, such as precise references to citation sources. Complex questions that require insights from extensive text corpora typically present significant challenges to the RAG paradigm. However, with the QA search, they can also be effectively addressed in advance.

5.3.4 Easy Setup, Update, High Efficiency and Stability

The QA search module is designed for seamless integration with RAG, and it can be initiated with an empty list of pre-known QA-pairs. Over time, answers generated by RAG can be evaluated and labeled by human and/or LLMs. Questions for which RAG fails to generate correct answers are collected. These questions, along with their (offline) prepared answers, are added to the QA search module's list for future semantic searches. If the list exceeds

the maximum token limit, it can be divided into manageable blocks. Additionally, questions that RAG can answer correctly may also be added to the list, as QA search typically requires less computational power and delivers shorter response time. Moreover, the answers returned by the QA search module remain consistent, enhancing the overall stability of the system's responses.

6 Conclusion

In this work, we introduced and demonstrated a straightforward QA search approach as an additional, integrable component to a traditional RAG pipeline, particularly for scenarios where the baseline RAG struggles to generate accurate answers for certain questions. Our empirical analysis shows that even with a simple setup, this approach can enhance performance and reduce the resource consumption of the conventional RAG paradigm. Furthermore, it enables applications to be implemented, updated, and scaled in an efficient and effective manner.

Limitations

This work has three main limitations. First, while we successfully designed experiments to compare our approach with RAG using the same datasets, we explored only a limited number of combinations of adjustable factors within the RAG pipeline. The baseline accuracy of a RAG approach can vary significantly depending on the dataset used and the specific implementation details throughout the entire RAG process. Second, we did not conduct an in-depth analysis of the incorrectly answered questions (by both RAG and QA search) to identify the precise reasons for these failures. Third, for the comparative experiments, we relied on LLMs to generate synthetic QA-pairs from two document sets, as the four QA datasets used in the feasibility experiments do not include corresponding text blocks. But we began our work using some of our internal data sets, which are not accessible to LLMs. Following the success of our experiments with these internal cases, we decided to publish this concept. To ensure data confidentiality, we utilized public data instead of our actual internal data for the publication. However, we achieved comparable results and accuracy using both the real internal data and the publicly available synthetic data/QA-pairs. Additionally, we depended on LLMs to evaluate the quality of all generated QA-pairs and were only able to manually verify 10% of them through random sampling.

Acknowledgments

Matthias Aßenmacher received funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the National Research Data Infrastructure – NFDI 27/1 - 460037581 - BERD@NFDI.

References

Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. 2023. Lift yourself up: Retrieval-augmented text generation with self memory. *Preprint*, arXiv:2305.02437.

Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. 2022. Promptagator: Few-shot dense retrieval from 8 examples. *Preprint*, arXiv:2209.11755.

Google DeepMind. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

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.

Alibaba Group. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *Preprint*, arXiv:2002.08909.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Atlas: Few-shot learning with retrieval augmented language models. *Preprint*, arXiv:2208.03299.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

Minki Kang, Jin Myung Kwak, Jinheon Baek, and Sung Ju Hwang. 2023. Knowledge graph-augmented

language models for knowledge-grounded dialogue generation. *Preprint*, arXiv:2305.18846.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. *Preprint*, arXiv:1911.00172.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. 2023a. Cmmlu: Measuring massive multitask language understanding in chinese. *Preprint*, arXiv:2306.09212.

Xinze Li, Zhenghao Liu, Chenyan Xiong, Shi Yu, Yu Gu, Zhiyuan Liu, and Ge Yu. 2023b. Structure-aware language model pretraining improves dense retrieval on structured data. *Preprint*, arXiv:2305.19912.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023c. Towards general text embeddings with multi-stage contrastive learning. *Preprint*, arXiv:2308.03281.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023a. Query rewriting for retrieval-augmented large language models. *Preprint*, arXiv:2305.14283.

Yubo Ma, Yixin Cao, Yong Hong, and Aixin Sun. 2023b. Large language model is not a good few-shot information extractor, but a good reranker for hard samples! In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics.

META. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Preprint*, arXiv:2302.00083.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *Preprint*, arXiv:2112.01488.

Leon Marius Schröder, Clemens Gutknecht, Oubada Alkiddeh, Susanne Weiß, and Leon Lukas. 2022. Lhm-dienstleistungen-qa - german public domain question-answering dataset.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *Preprint*, arXiv:2301.12652.

Noah A. Smith, Michael Heilman, and Rebecca Hwa. 2008. Question generation as a competitive undergraduate course project. In *The NSF Workshop on the Question Generation Shared Task and Evaluation Challenge, Arlington, VA, September 2008.*

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö. Arık. 2025. Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. *Preprint*, arXiv:2410.07176.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. *Preprint*, arXiv:2303.07678.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. *Preprint*, arXiv:2109.01652.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *Preprint*, arXiv:2310.04408.

Haoyan Yang, Zhitao Li, Yong Zhang, Jianzong Wang, Ning Cheng, Ming Li, and Jing Xiao. 2023. Prca: Fitting black-box large language models for retrieval question answering via pluggable reward-driven contextual adapter. *Preprint*, arXiv:2310.18347.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models. *Preprint*, arXiv:2310.06117.

Shengyao Zhuang, Bing Liu, Bevan Koopman, and Guido Zuccon. 2023. Open-source large language models are strong zero-shot query likelihood models for document ranking. *Preprint*, arXiv:2310.13243.

A Prompts

```
"""
You will be provided with question delimited by triple quotes. \
Please create {num_question} questions based on it with same meaning in {lang}, without quotes. \
Please list only the questions, no other text around. \
{text_length_spec}

\"\"\"{question}\"\"\"
"""
text_length_spec = "And it should not more than {question_length} characters."
```

Figure 6: Prompt for Rephrasing Question

```
"""
You will be provided with text delimited by triple quotes. \
It's a table of questions and answers, where answers are in the form of a reference number. \
If a question can be found in the table, return the reference number of the answer.

If the question cannot be found in the table, \
then simply return  \"-1\"

\"\"\"{questions_in_list}\"\"\"
"""
```

Figure 7: Prompt for Answering Question with New Approach

```
"""
Your task is to write a factoid question and an answer given a context.
Your factoid question should be answerable with a specific, concise piece of factual information from the context.
Your factoid question should be formulated in the same style as questions users could ask in a search engine.
This means that your factoid question MUST NOT mention something like "according to the passage" or "context".

Provide your answer as follows:

Output:::
Factoid question: (your factoid question)
Answer: (your answer to the factoid question)

Now here is the context.

Context: {context}\n
Output:::"""
```

Figure 8: Prompt for Generating Synthetic QA-pairs

```
question_groundedness_critique_prompt = """You will be given a context and a question.
Your task is to provide a 'total rating' scoring how well one can answer the given question unambiguously with the given context.
Give your answer on a scale of 1 to 5, where 1 means that the question is not answerable at all given the context, and 5 means that
the question is clearly and unambiguously answerable with the context.

Provide your answer as follows:

Answer:::
Evaluation: (your rationale for the rating, as a text)
Total rating: (your rating, as a number between 1 and 5)

You MUST provide values for 'Evaluation:' and 'Total rating:' in your answer.

Now here are the question and context.

Question: {question}\n
Context: {context}\n
Answer::: """
```

Figure 9: Prompt for Evaluating Generated QA-pairs with Critique Groundedness

```
"""You will be given a question.
Your task is to provide a 'total rating' representing how useful this question can be to machine learning developers building
NLP applications with the Hugging Face ecosystem.
Give your answer on a scale of 1 to 5, where 1 means that the question is not useful at all, and 5 means that the question is
extremely useful.

Provide your answer as follows:

Answer:::
Evaluation: (your rationale for the rating, as a text)
Total rating: (your rating, as a number between 1 and 5)

You MUST provide values for 'Evaluation:' and 'Total rating:' in your answer.

Now here is the question.

Question: {question}\n
Answer::: """
```

Figure 10: Prompt for Evaluating Generated QA-pairs with Critique Relevance

```
"""You will be given a question.
Your task is to provide a 'total rating' representing how context-independent this question is.
Give your answer on a scale of 1 to 5, where 1 means that the question depends on additional information to be understood, and 5 means that the question makes sense by itself.
For instance, if the question refers to a particular setting, like 'in the context' or 'in the document', the rating must be 1.
The questions can contain obscure technical nouns or acronyms like Gradio, Hub, Hugging Face or Space and still be a 5: it must simply be clear to an operator with access to documentation what the question is about.

For instance, "What is the name of the checkpoint from which the ViT model is imported?" should receive a 1, since there is an implicit mention of a context, thus the question is not independent from the context.

Provide your answer as follows:

Answer:::
Evaluation: (your rationale for the rating, as a text)
Total rating: (your rating, as a number between 1 and 5)

You MUST provide values for 'Evaluation:' and 'Total rating:' in your answer.

Now here is the question.

Question: {question}\n
Answer::: """
```

Figure 11: Prompt for Evaluating Generated QA-pairs with Critique Stand-alone

```
"""<|system|>
Using the information contained in the context,
give a comprehensive answer to the question.
Respond only to the question asked, response should be concise and relevant to the question.
Provide the number of the source document when relevant.
If the answer cannot be deduced from the context, do not give an answer.</s>
<|user|>
Context:
{context}
---
Now here is the question you need to answer.

Question: {question}
</s>
<|assistant|>
"""
```

Figure 12: Prompt for Answering Question with RAG

```
"""###Task Description:
An instruction (might include an Input inside it), a response to evaluate, a reference answer that gets a score of 5, and a score rubric representing a evaluation criteria are given.
1. Write a detailed feedback that assess the quality of the response strictly based on the given score rubric, not evaluating in general.
2. After writing a feedback, write a score that is an integer between 1 and 5. You should refer to the score rubric.
3. The output format should look as follows: \"Feedback: {{write a feedback for criteria}} [RESULT] {{an integer number between 1 and 5}}\"
4. Please do not generate any other opening, closing, and explanations. Be sure to include [RESULT] in your output.

###The instruction to evaluate:
{instruction}

###Response to evaluate:
{response}

###Reference Answer (Score 5):
{reference_answer}

###Score Rubrics:
[Is the response correct, accurate, and factual based on the reference answer?]
Score 1: The response is completely incorrect, inaccurate, and/or not factual.
Score 2: The response is mostly incorrect, inaccurate, and/or not factual.
Score 3: The response is somewhat correct, accurate, and/or factual.
Score 4: The response is mostly correct, accurate, and factual.
Score 5: The response is completely correct, accurate, and factual.

###Feedback:"""
```

Figure 13: Prompt for Evaluating Answers Generated with RAG

B Experiment Setup

Question	Answer	Rephr. Variant	Question rephr. by Model 1	Short Question rephr. by Model 1	...	Question rephr. by Model 9	Short Question rephr. by Model 9
Q1	A1	1	rephr. variant 1 to Q1 by model 1	rephr. short variant 1 to Q1 by model 1	...	rephr. variant 1 to Q1 by model 9	rephr. short variant 1 to Q1 by model 9
		2	rephr. variant 2 to Q1 by model 1	rephr. short variant 2 to Q1 by model 1	...	rephr. variant 2 to Q1 by model 9	rephr. short variant 2 to Q1 by model 9
		3	rephr. variant 3 to Q1 by model 1	rephr. short variant 3 to Q1 by model 1	...	rephr. variant 3 to Q1 by model 9	rephr. short variant 3 to Q1 by model 9
Q2	A2	1	rephr. variant 1 to Q2 by model 1	rephr. short variant 1 to Q2 by model 1	...	rephr. variant 1 to Q2 by model 9	rephr. short variant 1 to Q2 by model 9
		2	rephr. variant 2 to Q2 by model 1	rephr. short variant 2 to Q2 by model 1	...	rephr. variant 2 to Q2 by model 9	rephr. short variant 2 to Q2 by model 9
		3	rephr. variant 3 to Q2 by model 1	rephr. short variant 3 to Q2 by model 1	...	rephr. variant 3 to Q2 by model 9	rephr. short variant 3 to Q2 by model 9
...	...	...	...	...	...	...	...

Table 2: 54 rephrased expressions by 9 models to each original question

C Evaluation Results for Feasibility Experiments

Tables 3-6 below (one table per QA dataset) present questions rephrased by various LLMs, both without and with length limitations (denoted as "short"), arranged vertically. Horizontally, each rephrased question variant is answered by different LLMs. Each value represents the average accuracy across the entire sample set. Bolded values within the table indicate the LLM that achieves the highest answering performance for each question variant. Furthermore, the bolded average values in the bottom row represent the overall answering performance of each LLM across all question variants. These values are also depicted in Figure 2.

Question	**Answered By**									
Rephrased By	gpt35	gpt4o	llama3.1	llama3.3	qwen2	qwen2.5:7b	mixtral:8x7b	mixtral:8x22b	gemma2	gemma2:27b
original question	92.95%	**100.00%**	22.76%	87.80%	27.10%	75.34%	97.56%	**100.00%**	74.53%	38.75%
gpt35	84.28%	96.48%	21.95%	86.72%	40.92%	71.27%	**98.10%**	89.16%	89.16%	62.06%
gpt35 short	85.64%	94.31%	20.05%	86.72%	32.52%	71.54%	**94.85%**	90.79%	86.45%	55.83%
gpt4o	88.35%	98.92%	21.14%	84.28%	38.48%	71.54%	**99.19%**	96.21%	86.18%	51.22%
gpt4o short	85.09%	97.02%	22.49%	86.45%	34.96%	69.38%	**97.83%**	93.22%	82.93%	56.37%
llama3.1	80.76%	93.22%	20.33%	82.11%	34.69%	71.00%	**94.85%**	87.53%	82.11%	51.49%
llama3.1 short	72.63%	85.91%	19.78%	78.86%	32.52%	60.16%	**86.72%**	81.03%	77.78%	50.68%
qwen2	79.67%	95.66%	20.33%	74.80%	38.21%	70.73%	**95.93%**	88.35%	78.32%	41.73%
qwen2 short	72.09%	84.28%	18.16%	72.36%	35.23%	60.43%	**89.16%**	81.03%	76.42%	46.07%
qwen2.5:7b	79.67%	**95.39%**	21.14%	80.76%	35.50%	73.44%	94.58%	92.14%	82.38%	48.78%
qwen2.5:7b short	70.73%	84.01%	18.43%	75.07%	31.17%	64.23%	**85.91%**	78.59%	74.53%	48.51%
mixtral:8x7b	84.82%	95.93%	22.49%	82.93%	43.09%	74.25%	**97.56%**	93.77%	85.64%	49.32%
mixtral:8x7b short	84.01%	96.21%	21.41%	83.74%	38.21%	73.71%	**98.10%**	93.77%	85.91%	52.03%
mixtral:8x22b	83.20%	96.21%	21.95%	79.67%	39.57%	69.11%	**97.83%**	94.58%	83.47%	46.61%
mixtral:8x22b short	81.84%	88.89%	20.87%	88.62%	32.52%	68.29%	**95.12%**	91.60%	85.37%	46.88%
gemma2	84.28%	97.02%	22.49%	86.45%	36.59%	73.17%	**97.83%**	92.41%	85.09%	53.12%
gemma2 short	82.38%	92.14%	20.60%	88.35%	33.33%	71.27%	**92.95%**	89.97%	81.30%	53.39%
gemma2:27b	86.45%	96.75%	21.68%	83.20%	38.75%	70.46%	**98.10%**	92.14%	85.09%	54.20%
gemma2:27b short	79.95%	91.87%	20.60%	84.28%	36.31%	65.31%	**94.04%**	85.37%	79.40%	51.49%
Mean	**82.04%**	**93.70%**	**20.98%**	**82.80%**	**35.77%**	**69.72%**	**95.06%**	**90.09%**	**82.21%**	**50.45%**

Table 3: Detailed Answering Accuracy Results for QA Set OpenBookQA

Question	**Answered By**									
Rephrased By	gpt35	gpt4o	llama3.1	llama3.3	qwen2	qwen2.5:7b	mixtral:8x7b	mixtral:8x22b	gemma2	gemma2:27b
original question	98.58%	99.76%	35.93%	95.27%	77.54%	92.91%	**100.00%**	**100.00%**	92.20%	76.83%
gpt35	95.04%	85.11%	35.70%	88.65%	74.70%	89.13%	**95.98%**	87.94%	90.31%	74.94%
gpt35 short	94.56%	80.61%	38.53%	88.18%	69.98%	89.83%	**96.22%**	91.02%	89.60%	74.00%
gpt4o	96.69%	89.36%	33.81%	89.83%	71.87%	91.25%	**96.93%**	95.04%	91.25%	73.29%
gpt4o short	**96.45%**	82.03%	39.24%	91.02%	74.23%	90.78%	95.98%	93.85%	91.96%	76.12%
llama3.1	**94.56%**	84.87%	32.86%	85.82%	73.52%	88.18%	94.09%	87.71%	91.25%	75.89%
llama3.1 short	89.83%	78.96%	36.88%	84.63%	66.90%	84.40%	**90.78%**	87.23%	87.94%	77.07%
qwen2	90.78%	90.78%	30.26%	87.00%	75.41%	85.34%	**95.74%**	89.36%	88.89%	72.81%
qwen2 short	90.31%	84.40%	34.04%	83.45%	71.39%	88.65%	**95.51%**	85.58%	89.13%	72.34%
qwen2.5:7b	93.62%	91.25%	31.68%	88.18%	72.81%	88.18%	**96.22%**	90.07%	91.73%	74.00%
qwen2.5:7b short	93.38%	81.80%	35.70%	84.16%	75.41%	89.13%	**93.62%**	87.23%	90.07%	75.41%
mixtral:8x7b	94.80%	91.25%	33.33%	88.89%	74.70%	90.54%	**95.98%**	92.91%	90.78%	74.70%
mixtral:8x7b short	95.27%	91.02%	32.62%	90.07%	76.36%	88.65%	**95.98%**	93.85%	90.78%	76.12%
mixtral:8x22b	93.62%	91.73%	30.97%	88.65%	73.76%	89.60%	**96.45%**	91.25%	90.31%	73.05%
mixtral:8x22b short	91.96%	78.72%	35.93%	85.58%	72.10%	88.42%	**93.85%**	89.83%	89.36%	74.94%
gemma2	**95.27%**	88.89%	33.81%	87.71%	70.45%	90.54%	**95.27%**	92.43%	91.02%	75.89%
gemma2 short	90.31%	77.54%	33.81%	89.60%	66.19%	84.87%	**93.14%**	91.02%	90.78%	78.96%
gemma2:27b	93.85%	87.47%	32.15%	88.89%	74.70%	89.13%	**97.16%**	89.13%	90.07%	77.07%
gemma2:27b short	93.62%	82.74%	35.70%	89.60%	65.96%	87.47%	**93.85%**	89.60%	91.25%	77.78%
Mean	**93.82%**	**86.23%**	**34.37%**	**88.17%**	**72.53%**	**88.79%**	**95.41%**	**90.79%**	**90.46%**	**75.33%**

Table 4: Detailed Answering Accuracy Results for QA Set CMU-Wiki-QA

Question	**Answered By**									
Rephrased By	gpt35	gpt4o	llama3.1	llama3.3	qwen2	qwen2.5:7b	mixtral:8x7b	mixtral:8x22b	gemma2	gemma2:27b
original question	95.45%	99.24%	24.49%	89.90%	79.04%	91.16%	**100.00%**	**100.00%**	91.41%	83.59%
gpt35	94.19%	**94.70%**	24.75%	88.38%	75.51%	85.10%	93.18%	93.18%	89.65%	81.57%
gpt35 short	89.14%	**93.18%**	21.97%	86.62%	71.21%	79.29%	89.90%	87.12%	83.59%	78.28%
gpt4o	94.19%	**98.23%**	25.25%	91.16%	78.79%	89.90%	97.98%	95.71%	91.16%	82.07%
gpt4o short	83.84%	82.32%	24.75%	80.56%	69.44%	75.25%	**85.61%**	78.28%	78.54%	73.99%
llama3.1	**86.11%**	84.85%	23.74%	77.78%	69.19%	75.51%	**86.11%**	76.01%	79.80%	78.03%
llama3.1 short	70.96%	64.90%	22.47%	69.70%	63.38%	62.12%	**73.74%**	60.35%	65.91%	71.21%
qwen2	82.07%	79.55%	23.74%	78.54%	69.95%	71.46%	82.58%	75.76%	**83.84%**	78.79%
qwen2 short	**81.31%**	75.51%	27.78%	73.74%	67.93%	71.21%	79.29%	68.18%	78.28%	76.77%
qwen2.5:7b	85.61%	**86.62%**	23.23%	83.59%	69.95%	74.49%	86.11%	74.24%	84.85%	79.29%
qwen2.5:7b short	68.69%	61.87%	24.24%	**71.72%**	58.84%	63.13%	70.71%	60.86%	66.92%	65.66%
mixtral:8x7b	93.18%	**96.72%**	24.24%	89.39%	73.74%	83.84%	92.42%	90.91%	88.13%	80.30%
mixtral:8x7b short	88.89%	**92.17%**	24.49%	85.61%	77.53%	79.80%	87.63%	86.11%	85.10%	79.55%
mixtral:8x22b	90.15%	**92.93%**	24.75%	86.62%	75.76%	78.54%	90.40%	85.10%	85.86%	82.83%
mixtral:8x22b short	80.56%	**82.32%**	26.01%	80.30%	68.43%	66.16%	80.05%	71.97%	76.77%	80.05%
gemma2	91.41%	**95.20%**	25.51%	87.37%	76.26%	81.57%	91.67%	86.87%	86.36%	79.29%
gemma2 short	80.56%	76.26%	21.72%	80.56%	62.12%	67.17%	**81.06%**	77.78%	76.26%	76.26%
gemma2:27b	93.18%	**93.69%**	26.01%	88.38%	74.75%	80.05%	89.14%	89.14%	86.11%	80.30%
gemma2:27b short	76.52%	76.77%	21.46%	**79.55%**	61.62%	66.67%	75.76%	67.93%	75.00%	74.75%
Mean	**85.58%**	**85.63%**	**24.24%**	**82.60%**	**70.71%**	**75.92%**	**85.96%**	**80.29%**	**81.77%**	**78.03%**

Table 5: Detailed Answering Accuracy Results for QA Set LHM-Dienstleistungen-QA

Question	**Answered By**									
Rephrased By	gpt35	gpt4o	llama3.1	llama3.3	qwen2	qwen2.5:7b	mixtral:8x7b	mixtral:8x22b	gemma2	gemma2:27b
original question	97.20%	**100.00%**	21.91%	86.01%	48.72%	92.07%	92.07%	99.77%	87.18%	55.71%
gpt35	**94.17%**	93.01%	24.01%	90.21%	62.70%	88.81%	88.11%	80.19%	91.84%	69.00%
gpt35 short	90.68%	**91.61%**	28.21%	86.95%	55.48%	86.95%	78.32%	72.96%	89.28%	70.16%
gpt4o	96.97%	**97.20%**	24.94%	87.88%	57.58%	93.47%	91.84%	95.80%	88.58%	61.07%
gpt4o short	95.80%	**97.67%**	27.51%	92.07%	57.34%	93.01%	91.14%	93.01%	89.51%	67.83%
llama3.1	**84.62%**	80.65%	24.48%	73.89%	48.95%	79.02%	72.96%	67.83%	79.95%	63.87%
llama3.1 short	72.49%	71.79%	25.17%	73.66%	44.06%	69.23%	57.34%	50.12%	**74.13%**	62.47%
qwen2	90.21%	**91.38%**	24.94%	84.62%	60.84%	84.85%	79.02%	76.46%	87.88%	62.70%
qwen2 short	89.04%	**89.98%**	24.48%	86.95%	59.21%	87.18%	79.25%	67.37%	83.22%	70.40%
qwen2.5:7b	92.07%	**93.71%**	22.14%	89.51%	60.37%	92.54%	84.85%	78.79%	89.04%	68.07%
qwen2.5:7b short	86.71%	**89.04%**	26.57%	88.81%	52.91%	86.25%	75.99%	64.80%	87.41%	71.10%
mixtral:8x7b	85.78%	**92.77%**	23.54%	83.68%	57.34%	84.38%	80.42%	76.69%	85.78%	64.34%
mixtral:8x7b short	82.05%	**89.51%**	23.78%	83.22%	50.35%	81.82%	75.29%	71.10%	84.85%	63.17%
mixtral:8x22b	**92.77%**	92.07%	21.68%	84.85%	59.21%	88.34%	83.22%	82.98%	87.88%	63.17%
mixtral:8x22b short	91.38%	**92.77%**	24.48%	89.74%	54.08%	89.98%	78.09%	78.09%	89.51%	69.00%
gemma2	94.41%	**96.04%**	24.48%	88.34%	59.91%	89.98%	89.28%	83.45%	88.81%	65.97%
gemma2 short	**88.34%**	86.71%	23.78%	87.41%	50.12%	85.31%	75.76%	64.34%	86.01%	71.79%
gemma2:27b	93.01%	**96.27%**	24.94%	89.28%	57.34%	93.01%	87.18%	83.92%	89.28%	67.60%
gemma2:27b short	92.31%	**93.24%**	27.74%	88.81%	53.38%	88.11%	75.76%	72.96%	89.28%	71.33%
Mean	**90.00%**	**91.34%**	**24.67%**	**86.10%**	**55.26%**	**87.07%**	**80.84%**	**76.87%**	**86.81%**	**66.25%**

Table 6: Detailed Answering Accuracy Results for QA Set CMMLU

Doc Set	Model for QA Generation	Context Size	Mean GPU (MB)	Mean Memory (MB)	Mean CPU Usage (%)	Mean Elapsed Time (seconds/question)	Mean Accuracy (%)
it-tech-doc	chatgpt35	300	5	162.68	0.00%	1.41	98.67%
it-tech-doc	chatgpt35	600	5	162.68	1.33%	2.61	97.33%
it-tech-doc	chatgpt35	900	5	162.69	0.67%	3.07	97.33%
it-tech-doc	chatgpt35	1200	5	162.69	1.33%	5.08	96.67%
it-tech-doc	chatgpt35	1500	5	162.71	1.33%	5.99	99.33%
it-tech-doc	chatgpt4o	300	5	164.10	0.00%	1.06	96.67%
it-tech-doc	chatgpt4o	600	5	164.10	1.33%	2.50	98.67%
it-tech-doc	chatgpt4o	900	5	164.10	1.33%	3.91	94.67%
it-tech-doc	chatgpt4o	1200	5	164.10	0.67%	4.48	93.33%
it-tech-doc	chatgpt4o	1500	5	164.10	1.33%	7.28	93.33%
financial-news	chatgpt35	300	5	161.54	2.33%	1.33	85.27%
financial-news	chatgpt35	600	5	161.54	3.10%	2.71	83.72%
financial-news	chatgpt35	900	5	161.54	0.00%	2.74	79.07%
financial-news	chatgpt35	1200	5	161.54	0.78%	4.07	76.74%
financial-news	chatgpt35	1500	5	161.55	2.33%	5.47	76.74%
financial-news	chatgpt4o	300	5	164.83	2.00%	1.78	94.00%
financial-news	chatgpt4o	600	5	164.83	0.67%	2.49	94.00%
financial-news	chatgpt4o	900	5	164.83	0.67%	3.66	94.00%
financial-news	chatgpt4o	1200	5	164.83	1.33%	4.53	90.67%
financial-news	chatgpt4o	1500	5	164.83	1.33%	5.85	94.00%
	Mean	300	5	163.29	1.08%	1.39	93.65%
		600	5	163.29	1.61%	2.58	93.43%
		900	5	163.29	0.67%	3.35	91.27%
		1200	5	163.29	1.03%	4.54	89.35%
		1500	5	163.30	1.58%	6.15	90.85%

Table 7: Accuracy, Resource and Time Consumption Across Varying Total Numbers of Questions in the Question Catalog, with 150 Questions as QA Search Context and GPT4o as the Answering LLM

Doc Set	Model for QA Generation	Context Size	Mean GPU (MB)	Mean Memory (MB)	Mean CPU Usage (%)	Mean Elapsed Time (seconds/question)	Mean Accuracy (%)
it-tech-doc	chatgpt35	30	5	161.53	0.00%	4.72	89.33%
it-tech-doc	chatgpt35	50	5	161.53	2.00%	2.98	95.33%
it-tech-doc	chatgpt35	100	5	161.53	0.67%	1.80	96.67%
it-tech-doc	chatgpt35	150	5	161.53	2.67%	1.45	98.67%
it-tech-doc	chatgpt35	300	5	161.55	4.67%	1.68	98.67%
it-tech-doc	chatgpt4o	30	5	162.28	0.67%	3.43	96.00%
it-tech-doc	chatgpt4o	50	5	162.28	0.67%	2.45	96.00%
it-tech-doc	chatgpt4o	100	5	162.27	2.67%	1.44	98.00%
it-tech-doc	chatgpt4o	150	5	162.27	1.33%	1.02	96.00%
it-tech-doc	chatgpt4o	300	5	162.27	2.67%	0.80	99.33%
financial-news	chatgpt35	30	5	161.38	0.00%	4.48	75.19%
financial-news	chatgpt35	50	5	161.38	0.78%	2.23	77.52%
financial-news	chatgpt35	100	5	161.38	2.33%	1.52	79.84%
financial-news	chatgpt35	150	5	161.38	1.55%	1.40	86.05%
financial-news	chatgpt35	300	5	161.39	0.78%	1.20	85.27%
financial-news	chatgpt4o	30	5	162.34	0.67%	4.83	92.00%
financial-news	chatgpt4o	50	5	162.34	5.33%	3.20	93.33%
financial-news	chatgpt4o	100	5	162.34	1.33%	2.16	94.67%
financial-news	chatgpt4o	150	5	162.34	3.33%	1.71	94.67%
financial-news	chatgpt4o	300	5	162.34	1.33%	1.46	98.67%
	Mean	30	5	161.88	0.33%	4.36	88.13%
		50	5	161.88	2.19%	2.72	90.55%
		100	5	161.88	1.75%	1.73	92.29%
		150	5	161.88	2.22%	1.39	93.84%
		300	5	161.89	2.36%	1.28	95.48%

Table 8: Accuracy, Resource and Time Consumption Across Varying Number of Questions Used as QA Search Context, with 300 as Total Number of Questions in Question Catalog and GPT4o as the Answering LLM

D Evaluation Results for Comparative Experiments

D.1 Evaluation Results for Comparing RAG and RAG + QA Search

Test Case ID	Model for QA Generation	Chunk Size	If Rerank	Model for Answer Generation	Mean Accu. RAG	Mean Accu. RAG + QA Search
1	chatgpt35	200	TRUE	chatgpt4o	79.33%	100.00%
2	chatgpt35	200	FALSE	chatgpt4o	81.33%	100.00%
3	chatgpt35	400	TRUE	chatgpt4o	82.00%	99.33%
4	chatgpt35	400	FALSE	chatgpt4o	84.67%	100.00%
5	chatgpt35	200	TRUE	llama3.3	77.33%	99.33%
6	chatgpt35	200	FALSE	llama3.3	74.67%	99.33%
7	chatgpt35	400	TRUE	llama3.3	78.67%	100.00%
8	chatgpt35	400	FALSE	llama3.3	79.33%	100.00%
9	chatgpt4o	200	TRUE	chatgpt4o	82.55%	97.99%
10	chatgpt4o	200	FALSE	chatgpt4o	83.89%	98.66%
11	chatgpt4o	400	TRUE	chatgpt4o	82.67%	99.33%
12	chatgpt4o	400	FALSE	chatgpt4o	80.00%	100.00%
13	chatgpt4o	200	TRUE	llama3.3	79.33%	99.33%
14	chatgpt4o	200	FALSE	llama3.3	78.67%	97.33%
15	chatgpt4o	400	TRUE	llama3.3	76.67%	98.67%
16	chatgpt4o	400	FALSE	llama3.3	80.67%	99.33%
				Mean	80.11%	99.29%

Table 9: Mean Accuracy for RAG vs. RAG + QA Search for Document Set it-tech-doc

Test Case ID	Model for QA Generation	Chunk Size	If Rerank	Model for Answer Generation	Mean Accu. RAG	Mean Accu. RAG + QA Search
1	chatgpt35	200	TRUE	chatgpt4o	84.00%	96.67%
2	chatgpt35	200	FALSE	chatgpt4o	86.00%	98.00%
3	chatgpt35	400	TRUE	chatgpt4o	88.67%	97.33%
4	chatgpt35	400	FALSE	chatgpt4o	86.67%	98.00%
5	chatgpt35	200	TRUE	llama3.3	80.67%	94.00%
6	chatgpt35	200	FALSE	llama3.3	77.33%	93.33%
7	chatgpt35	400	TRUE	llama3.3	80.67%	96.00%
8	chatgpt35	400	FALSE	llama3.3	80.67%	96.00%
9	chatgpt4o	200	TRUE	chatgpt4o	76.67%	98.00%
10	chatgpt4o	200	FALSE	chatgpt4o	74.67%	96.00%
11	chatgpt4o	400	TRUE	chatgpt4o	71.33%	96.00%
12	chatgpt4o	400	FALSE	chatgpt4o	70.67%	96.67%
13	chatgpt4o	200	TRUE	llama3.3	74.67%	97.33%
14	chatgpt4o	200	FALSE	llama3.3	75.33%	97.33%
15	chatgpt4o	400	TRUE	llama3.3	71.33%	98.00%
16	chatgpt4o	400	FALSE	llama3.3	70.67%	98.00%
				Mean	78.13%	96.67%

Table 10: Mean Accuracy for RAG vs. RAG + QA Search for Document Set financial-news

D.2 Evaluation Results for Comparing QA Search and QA Search + RAG

Test Case ID	Model for QA Generation	Model for Answer Generation	Question Type	Mean Accu. QA Search	Mean Accu. QA Search + RAG
1	chatgpt35	chatgpt4o	normal	93.29%	99.33%
2	chatgpt35	chatgpt4o	original	100.00%	100.00%
3	chatgpt35	llama3.3	normal	99.33%	100.00%
4	chatgpt35	llama3.3	original	100.00%	100.00%
5	chatgpt4o	chatgpt4o	normal	90.00%	97.33%
6	chatgpt4o	chatgpt4o	original	100.00%	100.00%
7	chatgpt4o	llama3.3	normal	98.00%	99.33%
8	chatgpt4o	llama3.3	original	96.67%	99.33%
			Mean	97.16%	99.42%

Table 11: Mean Accuracy for QA Search vs. QA Search + RAG for Document Set it-tech-doc

Test Case ID	Model for QA Generation	Model for Answer Generation	Question Type	Mean Accu. QA Search	Mean Accu. QA Search + RAG
1	chatgpt35	chatgpt4o	normal	61.62%	91.92%
2	chatgpt35	chatgpt4o	original	75.76%	97.98%
3	chatgpt35	llama3.3	normal	71.72%	96.97%
4	chatgpt35	llama3.3	original	75.25%	96.97%
5	chatgpt4o	chatgpt4o	normal	79.33%	94.00%
6	chatgpt4o	chatgpt4o	original	100.00%	100.00%
7	chatgpt4o	llama3.3	normal	99.33%	99.33%
8	chatgpt4o	llama3.3	original	100.00%	100.00%
			Mean	82.88%	97.15%

Table 12: Mean Accuracy for QA Search vs. QA Search + RAG for Document Set financial-news

D.2.1 Evaluation Results for Time and Resource Consumption

Table 13 and 15 list average elapsed time per user query, as well as GPU, CPU, and memory consumption for the RAG approach for document sets it-tech-doc and financial-news, while Table 14 and 16 show the same metrics for the QA Search approach.

Test Case ID	Model for QA Generation	Chunk Size	If Rerank	Model for Answer Generation	Mean GPU (MB)	Mean Memory (MB)	Mean CPU Usage (%)	Mean Elapsed Time (seconds/question)
1	chatgpt35	200	TRUE	chatgpt4o	899.00	1,439.47	6.67%	9.64
2	chatgpt35	200	FALSE	chatgpt4o	899.00	1,453.43	6.00%	9.65
3	chatgpt35	400	TRUE	chatgpt4o	899.00	1,694.90	4.00%	9.84
4	chatgpt35	400	FALSE	chatgpt4o	899.00	1,696.27	4.67%	9.93
5	chatgpt35	200	TRUE	llama3.3	899.00	1,772.40	8.67%	13.76
6	chatgpt35	200	FALSE	llama3.3	899.00	1,772.99	6.67%	12.57
7	chatgpt35	400	TRUE	llama3.3	899.00	1,904.79	6.00%	15.38
8	chatgpt35	400	FALSE	llama3.3	899.00	1,672.34	4.67%	15.64
9	chatgpt4o	200	TRUE	chatgpt4o	899.00	1,787.59	8.67%	9.44
10	chatgpt4o	200	FALSE	chatgpt4o	899.00	1,729.71	6.00%	9.47
11	chatgpt4o	400	TRUE	chatgpt4o	899.00	1,799.73	7.33%	9.82
12	chatgpt4o	400	FALSE	chatgpt4o	899.00	1,800.57	2.67%	9.63
13	chatgpt4o	200	TRUE	llama3.3	899.00	1,860.98	3.33%	11.87
14	chatgpt4o	200	FALSE	llama3.3	899.00	1,861.81	7.33%	11.79
15	chatgpt4o	400	TRUE	llama3.3	899.00	1,803.70	7.33%	14.17
16	chatgpt4o	400	FALSE	llama3.3	899.00	1,804.46	4.67%	14.25
				Mean	899.00	1,740.95	5.92%	11.68

Table 13: Mean Resource & Time Consumption for RAG for Document Set it-tech-doc

Test Case ID	Model for QA Generation	Model for QA Search	Mean GPU (MB)	Mean Memory (MB)	Mean CPU Usage (%)	Mean Elapsed Time (seconds/question)
1	chatgpt35	chatgpt4o	6.00	160.29	3.36%	0.72
2	chatgpt35	llama3.3	6.00	160.60	1.34%	1.66
3	chatgpt4o	chatgpt4o	6.00	161.02	1.00%	0.58
4	chatgpt4o	llama3.3	6.00	161.68	2.67%	0.27
		Mean	6.00	160.90	2.09%	0.81

Table 14: Mean Resource & Time Consumption for QA Search for Document Set it-tech-doc

Test Case ID	Model for QA Generation	Chunk Size	If Rerank	Model for Answer Generation	Mean GPU (MB)	Mean Memory (MB)	Mean CPU Usage (%)	Mean Elapsed Time (seconds/question)
1	chatgpt35	200	TRUE	chatgpt4o	899.00	1,973.38	6.98%	9.44
2	chatgpt35	200	FALSE	chatgpt4o	899.00	1,974.33	3.88%	9.31
3	chatgpt35	400	TRUE	chatgpt4o	899.00	1,832.66	5.43%	9.42
4	chatgpt35	400	FALSE	chatgpt4o	899.00	1,833.28	3.10%	9.46
5	chatgpt35	200	TRUE	llama3.3	899.00	1,870.54	3.10%	12.77
6	chatgpt35	200	FALSE	llama3.3	899.00	1,871.24	6.20%	12.07
7	chatgpt35	400	TRUE	llama3.3	899.00	1,983.69	5.43%	13.44
8	chatgpt35	400	FALSE	llama3.3	899.00	1,984.38	7.75%	12.77
9	chatgpt4o	200	TRUE	chatgpt4o	899.00	1,857.77	5.33%	9.31
10	chatgpt4o	200	FALSE	chatgpt4o	899.00	1,858.87	11.33%	9.29
11	chatgpt4o	400	TRUE	chatgpt4o	899.00	1,771.13	3.33%	9.38
12	chatgpt4o	400	FALSE	chatgpt4o	899.00	1,688.79	4.00%	9.38
13	chatgpt4o	200	TRUE	llama3.3	899.00	1,857.64	10.00%	11.79
14	chatgpt4o	200	FALSE	llama3.3	899.00	1,846.05	4.00%	11.77
15	chatgpt4o	400	TRUE	llama3.3	899.00	1,814.63	8.67%	12.61
16	chatgpt4o	400	FALSE	llama3.3	899.00	1,815.73	4.67%	12.62
				Mean	899.00	1,864.63	5.82%	10.93

Table 15: Mean Resource & Time Consumption for RAG for Document Set financial-news

Test Case ID	Model for QA Generation	Model for QA Search	Mean GPU (MB)	Mean Memory (MB)	Mean CPU Usage (%)	Mean Elapsed Time (seconds/question)
1	chatgpt35	chatgpt4o	6.00	162.10	2.00%	0.95
2	chatgpt35	chatgpt4o	6.00	162.44	2.00%	0.58
3	chatgpt35	llama3.3	6.00	162.61	2.00%	0.98
4	chatgpt35	llama3.3	6.00	162.61	2.00%	0.41
		Mean	6.00	162.44	2.00%	0.73

Table 16: Mean Resource & Time Consumption for QA Search for Document Set financial-news