ITERKEY: Iterative Keyword Generation with LLMs for Optimizing Retrieval Augmented Generation

Anonymous ACL submission

Abstract

This paper introduces ITERKEY, an iterative keyword generation method that leverages large language models (LLMs) to optimize the retrieval-augmented generation (RAG) relying on sparse retrieval by keywords. Inspired by recent advancements showing that LLMs can manage, control, and optimize complex workflows in fields such as business process management and code generation, we aim to explore their potential in optimizing the entire RAG process. ITERKEY comprises three stages, all of which are performed by an LLM: keyword generation, answer generation, and answer verification. First, the LLM generates keywords that characterise the initial query. Then, it generates an answer based on the documents retrieved by the keywords. Finally, the LLM verifies the answer, and if it is inaccurate or incomplete, it refines the query with new keywords and iterates the whole process. Experimental results on four benchmark datasets show that ITERKEY notably improves retrieval accuracy and answer quality, with gains of up to 20% over baseline methods. These results confirm the effectiveness of our approach and highlight the potential of LLMs to optimize the RAG process through iterative keyword refinement and self-evaluation.

1 Introduction

Large Language Models (LLMs) (Touvron et al., 2023; OpenAI, 2023; Chiang et al., 2023; Dubey et al., 2024; Abdin et al., 2024; Team et al., 2024) have shown remarkable capabilities across various natural language processing tasks. However, they still face challenges like hallucinations, outdated knowledge, and challenges in handling complex queries requiring multi-hop reasoning or external information sources (Kandpal et al., 2023; Zhang et al., 2023b; Gao et al., 2024). These issues are particularly prominent in knowledge-intensive tasks (Lee et al., 2019; Zellers et al., 2018), where the necessary information may not be fully captured within the model's parameters.

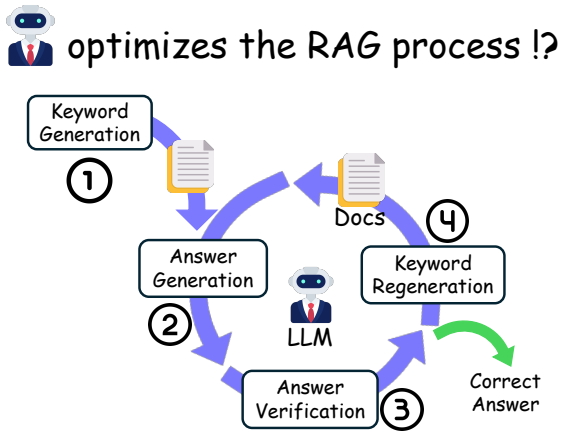

Figure 1: Leveraging the advanced reasoning capabilities of LLMs to optimize the entire RAG process.

Retrieval Augmented Generation (RAG) improves the accuracy and relevance of generated responses by integrating external knowledge, making it particularly effective for open-domain question answering (Lewis et al., 2020; Izacard and Grave, 2021; Shuster et al., 2021; Ram et al., 2023; Izacard et al., 2024). While improvements to individual RAG components, such as query expansion and retrieval tuning, have been explored (Gao et al., 2024; Fan et al., 2024), RAG's effectiveness heavily relies on the interaction between retrieval and generation. Often, these components function independently, leading to a lack of coherence and synergy (Jiang et al., 2022; Zhang et al., 2023a; Shao et al., 2023; Feng et al., 2024; Yu et al., 2023; Cheng et al., 2023; Shi et al., 2024).

To address this challenge, we propose **IterKey**: **Iter**ative **Key**word Generation with LLMs, a novel approach that leverages LLMs to optimize the entire RAG process through iterative keyword refinement and self-evaluation. By allowing the LLM to manage both retrieval and generation compo-

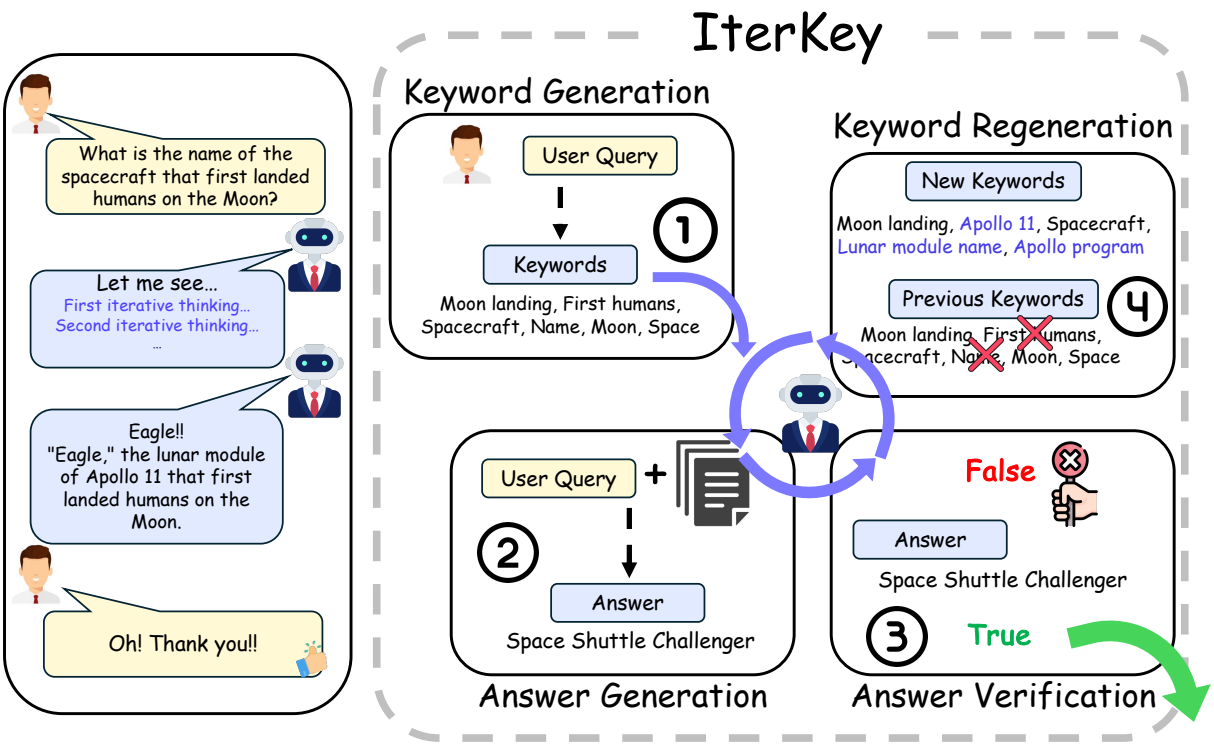

Figure 2: Overview of the IterKey process: Iterative keyword generation, answer generation, and verification using an LLM. The process refines keywords and answers until the correct response is found, enhancing RAG accuracy.

nents, ITERKEY enhances the interaction between them, resulting in more coherent and accurate responses. In ITERKEY, the RAG process is divided into three phases: keyword generation, answer generation, and answer verification phases—inspired by prior work (Creswell et al., 2023)—enabling the LLM to automatically optimize the workflow (Figure 1). An LLM generates critical keywords from the initial query by understanding both the BM25-based retrieval algorithm and the query's intent, ensuring that essential keywords, including implicit ones, are identified for effective document retrieval. These generated keywords are then used for retrieval, enhancing the retrieval process by incorporating both surface-level and implicit information. Subsequently, the LLM generates an answer based on the retrieved documents and verifies its correctness. If the answer is unsatisfactory, the process iterates, refining the keywords to better capture the user's intent and improve retrieval accuracy, until an accurate response is obtained.

We conducted experiments on four open-domain question answering (QA) datasets. The results demonstrate that ITERKEY significantly improves both retrieval accuracy and answer quality, achieving gains of up to 20% compared to baseline methods. These improvements validate the effectiveness of our approach and underscore the potential of using LLMs to holistically optimize the RAG process through iterative keyword refinement and self-evaluation.

2 ITERKEY: Iterative Keyword Generation with LLMs

Sparse retrieval algorithms, such as BM25 (Robertson and Zaragoza, 2009), are widely used for document search because of their efficiency handling large datasets. These algorithms rank documents based on keyword frequency and uniqueness, ensuring high visibility and interpretability by clearly showing which keywords influence the results. However, they struggle to capture the nuances and implicit information in the query, presenting limitations compared to dense retrieval methods.

ITERKEY leverages recent advancements in LLMs to manage and optimize the entire RAG process. By enabling the LLM to oversee both retrieval and generation components, ITERKEY aims to create a more unified and effective system. Specifically, we leverage the LLM's self-evaluation capabilities (Wang et al., 2024b) to continuously improve both retrieval and generation steps, ultimately enhancing the quality of the responses.

Step	Prompt
Step 1: Initial Keyword Generation	**System:** You are an assistant that generates keywords for information retrieval. **User:** Generate a list of important keywords related to the **query { q }**. Focus on keywords that are relevant and likely to appear in documents for BM25 search in the RAG framework. Output the keywords as: ["keyword1", "keyword2", "keyword3", ...]. Separate each keyword with a comma and do not include any additional text.
Step 2: Answer Generation using RAG	**System:** You are an assistant that generates answers based on retrieved documents. **User:** Here is a question that you need to answer: **Query: { q }** Below are some documents that may contain information relevant to the question. Consider the information in these documents while combining it with your own knowledge to answer the question accurately. Documents: { D } Provide a clear and concise answer. Do not include any additional text.
Step 3: Answer Validation	**System:** You are an assistant that validates whether the provided answer is correct. **User:** Is the following answer correct? **Query: { q }** Answer: { a } Respond 'True' or 'False'. Do not provide any additional explanation or text.
Step 4: Keyword Regeneration	**System:** You refine keywords to improve document retrieval for BM25 search in the RAG framework. **User:** Refine the keyword selection process to improve the accuracy of retrieving documents with the correct answer. **Query: { q }** Previous Keywords: { K } Provide the refined list of keywords in this format: ["keyword1", "keyword2", ...]. Separate each keyword with a comma and do not include any additional text.

Table 1: The prompts used in the IterKey for iterative refinement of keyword generation and answer validation.

IterKey enhances the entire RAG process by iteratively refining keyword selection to improve sparse retrieval. The process involves an interplay between keyword generation, document retrieval, answer generation, and answer verification, all orchestrated by an LLM. By refining key terms and evaluating the results at each step, IterKey identifies both explicit and implicit keywords, overcoming the limitations of sparse retrieval while maintaining visibility and interpretability to ensure more accurate and relevant results. Algorithm 1 shows the detail procedure and the actual prompts used in each step are shown in Table 1 with corresponding examples in Figure 2.

Step 1: Keyword Generation Given a user query q, the LLM is prompted to generate a set of critical keywords $\mathcal{K}^0$ that are essential for retrieving relevant documents. This step aims to capture the nuances and essential terms that traditional retrieval methods might overlook. In our example, for the query *"What is the name of the spacecraft that first landed humans on the Moon?"*, LLM generates important keywords like *Moon landing, Spacecraft, First humans*.

Algorithm 1 IterKey: Iterative Keyword Generation with LLMs

1: Initialize iteration counter $i = 0$
2: **Keyword Generation:** Generate critical keywords $\mathcal{K}^i$
3: **repeat**
4: **Answer Generation:** Retrieve documents $\mathcal{D}$ using $q + \mathcal{K}^i$ and generate answer a^i.
5: **Answer Validation:** If answer a^i is correct, **break**
6: **Keyword Regeneration:** If answer a^i is incorrect, refine keywords as $\mathcal{K}^{i+1}$
7: $i \leftarrow i + 1$
8: **until** Correct answer is found or maximum iterations N is reached
9: **Return** a^i

Step 2: Answer Generation using RAG The original query q is expanded with the generated keywords $\mathcal{K}^i$ to form an enhanced query $q + \mathcal{K}^i$. This expanded query is used to retrieve a set of top-k documents $\mathcal{D}^i = \{d_1, d_2, \ldots, d_k\}$ from an external corpus using a BM25-based retriever. Using the retrieved documents, LLM generates an

answer a^i to the original query q. E.g., LLM generates the answer *Space Shuttle Challenger* using the generated keywords and query.

Step 3: Answer verification In this step, LLM verifies whether the generated answer is correct based on the retrieved documents, ensuring that the RAG process has been executed correctly. LLM is required to respond with either 'True' or 'False'. If the LLM responds with 'True', it indicates that the answer is correct, the process concludes, and a^i is returned as the final answer. Upon reviewing the documents, LLM determines that *Space Shuttle Challenger* is incorrect and returns 'False' in our example.

Step 4: Keyword Regeneration If the LLM responds with 'False' in the validation step, indicating that the correct answer is not correct, we regenerate keywords. LLM is prompted to generate a new set of keywords $\mathcal{K}^{i+1}$ to improve the retrieval. The new keywords $\mathcal{K}^{i+1}$ are then used to form a new expanded query $q + \mathcal{K}^{i+1}$, and Steps 2 to 4 are repeated. This iterative process continues until the validation step returns 'True' or a predefined maximum number of iterations is reached. E.g., after receiving 'False', LLM regenerates new keywords like *Apollo 11, Lunar module name*, and the correct answer *Eagle* is finally retrieved in Figure 2.

3 Experimental Setup

3.1 Evaluation Methodology

We evaluated ITERKEY using four open-domain question answering datasets, i.e., Natural Questions (NQ) (Kwiatkowski et al., 2019), EntityQA (Li et al., 2019), WebQA (Chang et al., 2022), and HotpotQA (Yang et al., 2018). Following prior work (Trivedi et al., 2023; Jiang et al., 2023b; Feng et al., 2024), we randomly sampled 500 entries from each dataset due to the computational cost of experiments. Our method was evaluated under zero-shot settings.

Generated answers are evaluated using the standard Exact Match (EM) metric (Rajpurkar et al., 2016). Under this framework, a response is considered correct if it precisely matches any of the reference answers after a standard normalization process. This normalization entails converting all generated text to lowerE.g.,, removing articles, eliminating punctuation, and consolidating multiple consecutive whitespace characters into single spaces.

Additionally, to analyze the accuracy of query expansion via keyword augmentation, we measure recall by checking if the retrieved documents contain the correct answer. This offers a more detailed evaluation of retrieval performance. We used the December 2018 Wikipedia dump as the retrieval corpus for all datasets(Izacard et al., 2024), chunking the text into segments of 256 tokens. To preserve context, an overlap of 50 tokens was applied at the beginning and end of each chunk.

3.2 Models

Retrieval Models We utilized BM25 (Robertson and Zaragoza, 2009), implemented using BM25S (Lù, 2024)[1], as the base retriever.

LLMs In this study, we selected four LLMs: Llama-3.1-(8B, 70B) (Dubey et al., 2024), Gemma-2 (Team et al., 2024), and Phi-3.5-mini (Abdin et al., 2024). These models were chosen based on preliminary experiments with several candidate models, following two main criteria to automatically implement our method (Section 2) without human intervention:

1. **Adherence to Specified Output Format:** The model must follow the designated generation format. In keyword generation and verification, it should produce outputs that conform to the specified format with minimal post-processing, generating lists without unnecessary extra text.

2. **Accurate Understanding of Instructions:** The model must accurately comprehend the intent of the instructions. It should generate outputs based on a correct understanding of the instructions. During verification and keyword regeneration, it should produce consistent outputs aligned with the task requirements.

Models such as Llama 2 (Touvron et al., 2023), Quen 1.5 (Bai et al., 2023), Vicuna (Chiang et al., 2023), and Mistral (Jiang et al., 2023a) were not adopted for ITERKEY as they either produced erroneous outputs or failed to adhere to the specified format, making them unsuitable for use in this context. Additionally, all selected models received instruction tuning, as models without instruction tuning were unable to generate controlled outputs effectively.

[1] https://github.com/xhluca/bm25s

Method	Model	Size (B)	**Entity QA**	**HotpotQA**	**Natural QA**	**WebQA**
Baseline	Llama-3.1	8B	33.6	31.2	40.6	53.4
	Llama-3.1	70B	45.2	41.4	46.0	54.0
	Gemma-2	9B	10.6	11.6	9.2	20.8
	Phi-3.5-mini	3.8B	24.6	25.4	25.8	44.0
Baseline RAG	Llama-3.1	8B	28.0	31.4	42.0	55.8
	Llama-3.1	70B	34.4	34.2	40.8	50.8
	Gemma-2	9B	14.2	20.4	31.8	43.0
	Phi-3.5-mini	3.8B	19.0	23.4	29.2	45.2
One-Step Keyword	Llama-3.1	8B	49.2	41.2	47.6	54.2
	Llama-3.1	70B	**55.0**	42.2	49.6	53.2
	Gemma-2	9B	18.2	16.0	15.2	17.4
	Phi-3.5-mini	3.8B	37.2	32.6	32.2	42.4
IterKey	Llama-3.1	8B	53.6	**47.8**	**54.8**	**59.8**
	Llama-3.1	70B	52.0	47.0	52.6	55.6
	Gemma-2	9B	7.2	7.6	8.1	13.8
	Phi-3.5-mini	3.8B	41.8	33.6	35.2	49.4

Table 2: 'Baseline' shows results without retrieval, while 'Baseline RAG' includes retrieval based on the question. 'One-Step Keyword ' uses one generated keyword for retrieval. 'IterKey,' our proposed method, iteratively generates keywords, repeating the process up to a maximum of five times to refine retrieval for optimal answers.

4 Results

The main results are shown in Table 2. For the Llama-3.1-8B and Phi-3.5-mini Llama-3.1-70B models, ITERKEY improved the accuracy over 'Baseline' (without RAG) across all tasks, with gains from 10% to 20%. For example, in Entity QA, the Llama-3.1-8B model saw a 20% increase in accuracy, from 33.6% to 53.6%, and the Phi-3.5-mini model also showed an improvement of about 17%, from 24.6% to 41.8%. Furthermore, notable gains were observed in Natural QA for both the Llama-3.1-8B and Phi-3.5-mini models, with ITERKEY proving particularly effective in smaller models. On the other hand, in the E.g., of the Gemma-2 model, accuracy decreased across all methods, and applying ITERKEY further reduced accuracy by 7.2%. In contrast, the 'Baseline RAG' method improved accuracy by 14.2%, indicating that the Gemma-2 model may not adequately handle ITERKEY's iterative keyword generation process.

Comparing different sizes of the Llama-3.1 model revealed that, while the baseline Llama-3.1-70B achieved 10% higher accuracy across all tasks, the Llama-3.1-8B model, with one-tenth the number of parameters, performed better under ITERKEY. This suggests ITERKEY is more effective for small-scale models, while improvements are less pronounced in larger models, likely due to their extensive existing knowledge, limiting further enhancements through ITERKEY.

In comparing keyword methods, the One-Step Keyword approach improved accuracy over the baseline in most scores. For example, the Llama-3.1-8B model achieved 49.2% in Entity QA, and the Phi-3.5-mini model achieved 37.2%. This shows that using LLM-generated keywords for document retrieval can enhance accuracy. The keyword expansion itself holds value, suggesting that when humans search for information on the web, algorithms like BM25 or TF-IDF, using LLM-generated keywords may increase the likelihood of finding better answers. In other words, LLMs can grasp the RAG process and strategically select keywords to effectively retrieve information.

5 Analysis

5.1 LLM's Keyword Generation Ability

In this section, we examine the effectiveness of the keyword generation step in ITERKEY by analyzing recall rates and evaluating the impact of keyword quality on overall performance.

Recall Rate Analysis Figure 3 shows recall rates (%) for each model and number of iterations in

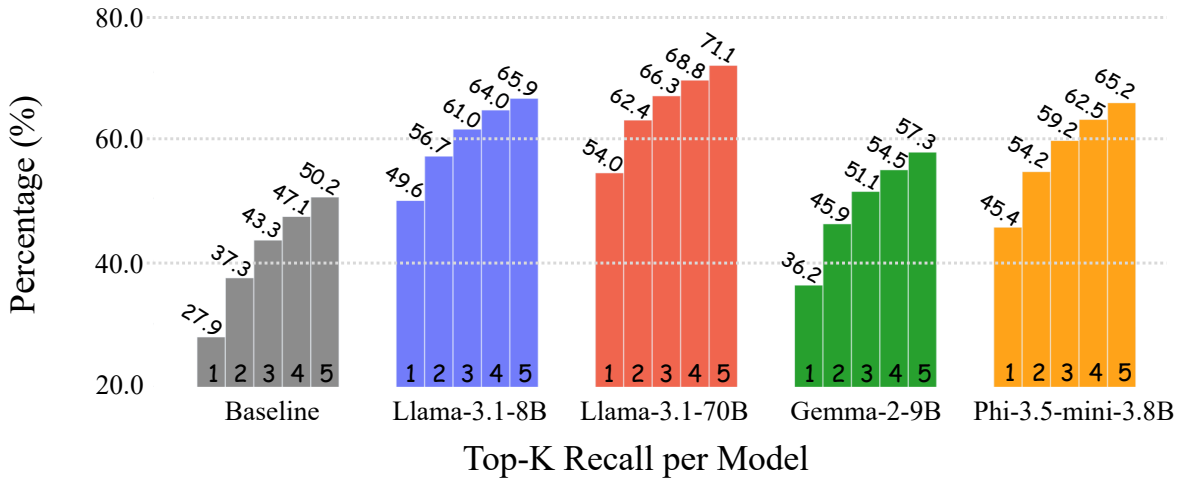

Figure 3: Top-K retrieval recall rates across models. Each bar shows the average percentage of documents with the correct answer retrieved over four tasks. Baseline represents standard query-based retrieval, while other models utilize Iterkey, with recall rates averaged over five iterations.

Method	Model	Size (B)	**Entity QA**	**HotpotQA**	**Natural QA**	**WebQA**
IterKey	Llama-3.1	8B	53.6	47.8	54.8	59.8
	Llama-3.1	70B	52.0	47.0	52.6	55.6
	Gemma-2	9B	7.2	7.6	8.1	13.8
	Phi-3.5-mini	3.8B	41.8	33.6	35.2	49.4
IterKey (ablation)	Llama-3.1	8B	55.4	45.6	53.8	58.8
	Llama-3.1	70B	52.0	47.0	52.6	55.6
	Gemma-2	9B	16.2	12.8	17.8	23.8
	Phi-3.5-mini	3.8B	45.6	34.8	41.0	47.8

Table 3: 'ITERKEY (ablation)' uses the high-quality keywords from the Llama-3.1-70B model, which had the highest recall, and applies them to three other models with 10 times fewer parameters, measuring accuracy using the ITERKEY method under the same settings.

ITERKEY. The recall rate refers to the proportion of retrieved documents containing the correct answer for the QA tasks. All models demonstrate higher recall rates than the baseline, indicating that keyword expansion by LLMs is functioning effectively. In particular, the Llama-3.1-70B model achieves a Top-1 recall rate equivalent to the baseline's Top-5, confirming that it is the best model for keyword generation in ITERKEY. However, considering the EM scores from the previous section, where overall accuracy with ITERKEY was better for the Llama-3.1-8B model, it suggests that although the Llama-3.1-70B model can generate high-quality keywords, its reasoning and verification abilities may limit performance improvement in the answer generation and verification steps. On the other hand, the Gemma-2 model shows lower recall rates and accuracy with ITERKEY compared to other models. This suggests Gemma-2 may not be suitable for ITERKEY, as it struggles with keyword generation and iterative refinement. Additionally, since the EM score with ITERKEY is worse than the baseline despite having a higher recall rate than the baseline RAG, it indicates that there is a problem in the verification step as well.

Impact of Keyword Quality on Performance Based on these results, we conducted additional experiments to further investigate the impact of keyword quality on ITERKEY's performance. Specifically, we used the high-quality keywords generated by the Llama-3.1-70B model to run ITERKEY with other models. Only the keyword generation step was performed by Llama-3.1-70B, while other steps were handled by each model. The results in Table 3 show little change in the performance of the

Setting	Model	Size (B)	**Entity QA**	**HotpotQA**	**Natural QA**	**WebQA**
Base	Llama-3.1	8B	53.6	47.8	54.8	59.8
	Llama-3.1	70B	**52.0**	**47.0**	**52.6**	**55.6**
	Gemma-2	9B	7.2	7.6	8.1	13.8
	Phi-3.5-mini	3.8B	41.8	33.6	35.2	49.4
VerifiedTrue	Llama-3.1	8B	59.6	52.9	59.9	64.8
	Llama-3.1	70B	**58.0**	**57.6**	**55.6**	**58.6**
	Gemma-2	9B	9.6	9.3	10.8	15.8
	Phi-3.5-mini	3.8B	46.5	38.9	40.7	53.4
VerifiedAll	Llama-3.1	8B	62.6	56.6	64.0	69.0
	Llama-3.1	70B	**66.2**	**60.0**	**65.2**	**69.8**
	Gemma-2	9B	20.0	19.8	19.8	23.8
	Phi-3.5-mini	3.8B	49.8	42.0	43.6	54.6

Table 4: Accuracy results for different models and settings across four QA tasks. **Setting Descriptions**: 'Base' is the basic setting where the first occurrence of "True" in the Answer Verification step is taken as the answer, and the iteration stops. 'VerifiedTrue' checks if any of the iterations judged as "True" contains the correct answer. 'VerifiedAll' considers all iteration results, including both "True" and "False" judgments, and checks if the correct answer is present in any of them.

Llama-3.1-8B model. This confirms that the gap between the EM score and recall in ITERKEY with the Llama-3.1-8B model is mainly due to the answer generation and verification steps. In contrast, the Phi-3.5-mini and Gemma-2 models showed improved accuracy when using the high-quality keywords generated by the Llama-3.1-70B model. The Phi-3.5-mini model showed improvements of up to 5%, while Gemma-2 showed up to a 10% increase in accuracy. These results indicate that keyword quality significantly impacts the performance of ITERKEY in these models, suggesting that even models with limited keyword generation capabilities have room for performance improvement by using high-quality keywords. Overall, these results highlight the importance of keyword quality in ITERKEY. Even for models with insufficient keyword generation capabilities, it is possible to improve performance by combining them with a stronger model that generates high-quality keywords.

5.2 LLM's Answer Verification Ability

In this section, we conducted additional experiments to evaluate the Answer Verification step in ITERKEY by iterating up to five times, even after a 'True' judgment. We compared accuracy across three settings to assess the verification performance of each model.

Comparing 'VerifiedAll' with 'Base' shows a 5–15% decrease in accuracy across all models. Notably, the Llama-3.1-70B model exhibits a difference of about 15% in all tasks, while Phi-3.5-mini shows the lowest error rate, with a decrease limited to 5–8% (see Table 4). By comparing 'Base' with 'VerifiedTrue', we can observe the rate of incorrectly answering 'True' when it should be 'False' (miss True rate). Similarly, comparing 'VerifiedTrue' with 'VerifiedAll' allows us to assess the rate of overlooking 'False' when it should be 'True' (miss False rate). Specifically, for Llama-3.1-8B, Llama-3.1-70B, and Phi-3.5-mini, the miss True rate is relatively higher than the miss False rate, indicating that these models often recognize incorrect answers as correct, reducing accuracy. On the other hand, Gemma-2 shows a higher miss False rate. These results show that the Llama-3.1 series contributes to the decrease in accuracy due to misjudgments in the Verification step. In contrast, Phi-3.5-mini exhibits the lowest error rate, confirming it as the most suitable model for Verification.

6 Related Work

LLMs for Process Optimization Recent LLMs are not only capable of generating text but also of planning, controlling, and optimizing complex processes (Shen et al., 2023; Wang et al., 2024a; Vidgof et al., 2023; Cui et al., 2024). By leveraging their powerful reasoning and inference capabilities, LLMs can manage and orchestrate various

components within a system, improving overall performance. For instance, in business process management (Vidgof et al., 2023) and code generation (Wang et al., 2024a), LLMs have been employed to plan and optimize workflows, showing their potential beyond text generation (Shen et al., 2023).

Retrieval Augmented Generation (RAG) RAG enhances response accuracy and relevance by incorporating external knowledge, making it effective for open-domain question answering (Lewis et al., 2020; Izacard and Grave, 2021; Shuster et al., 2021; Ram et al., 2023; Izacard et al., 2024). It also addresses the limitations of in-context learning, which struggles with complex questions and outdated information (Liu et al., 2022; Petroni et al., 2019; Borgeaud et al., 2022; Roberts et al., 2020). Recent RAG research has focused on improving components like query expansion, retrieval tuning, document summarization, and reranking (Gao et al., 2024; Fan et al., 2024). RAG's effectiveness depends on both its components and their interaction (Jiang et al., 2022; Zhang et al., 2023a). Optimizing retrieval, generation, and their interplay is key for enhancing overall performance and coherence. However, since these processes often function independently, maintaining coherence is challenging (Shi et al., 2024; Guo et al., 2023). Even if relevant documents are retrieved, they may not be fully utilized in generating responses (Ram et al., 2023; Dai et al., 2023; Xu et al., 2024; Asai et al., 2024). When retrieval falls short, the response can become irrelevant (Jiang et al., 2023b; Guo et al., 2023). This lack of synergy remains a key issue for RAG systems (Shao et al., 2023; Feng et al., 2024; Yu et al., 2023; Cheng et al., 2023).

Query Expansion and Document Expansion Query expansion techniques have advanced with LLMs, moving beyond traditional methods like relevance feedback and thesaurus-based expansion. Neural approaches now leverage LLMs' generative abilities, such as using sequence-to-sequence models to generate queries (Nogueira et al., 2019). However, these models can produce "hallucinated" queries, reducing retrieval effectiveness (Zhang et al., 2023b). Recent methods generate pseudo-documents from user queries to enhance retrieval in both sparse and dense systems (Wang et al., 2023). Similarly, generating hypothetical documents has improved zero-shot dense retrieval without relevance labels (Gao et al., 2023). These techniques highlight LLMs' ability to address challenges short or ambiguous queries by providing relevant contextual information (Zhang et al., 2023b; Asai et al., 2023; Zhou et al., 2023).

Iterative and Recursive Retrieval Iterative retrieval methods refine queries through intermediate reasoning to handle complex, multi-step information needs (Trivedi et al., 2023; Shao et al., 2023; Kim et al., 2023; Feng et al., 2024; Sun et al., 2023; Jiang et al., 2023b). These methods combine chain-of-thought (CoT) reasoning with retrieval, dynamically updating queries to improve retrieval accuracy and QA performance. Generative reasoning is also interleaved with retrieval, allowing for iterative refinement of both queries and responses to handle complex queries effectively (Yao et al., 2023; Arora et al., 2023; Wang et al., 2022; Jiang et al., 2023b).

Chain-of-Thought Retrieval Integrating Chain-of-Thought (CoT) reasoning with retrieval improves multi-step, knowledge-intensive tasks (Trivedi et al., 2023; Shao et al., 2023; Creswell et al., 2023; Zhou et al., 2023). CoT-guided retrieval enhances document relevance and factual accuracy while reducing hallucinations (Yao et al., 2023; Wei et al., 2022; Kojima et al., 2024; Zheng et al., 2024). Supervised iterative retrieval shows promise in multi-hop QA (Das et al., 2019; Xiong et al., 2021), while unsupervised approaches offer multi-step retrieval without extensive labeled data, suitable for few-shot and zero-shot scenarios (Khattab et al., 2023; Qi et al., 2019). Contrastive learning also improves unsupervised dense retrieval without labeled data (Gao and Callan, 2021).

7 Conclusion

In this study, we proposed ITERKEY, an iterative keyword generation framework that optimizes the entire RAG process using LLMs. By integrating keyword generation, answer generation, and answer verification in an LLM-driven loop, ITERKEY enhances synergy between retrieval and generation, resulting in more accurate and coherent responses. Experiments on four open-domain question answering datasets showed that ITERKEY improves retrieval accuracy and answer quality by up to 20% compared to baseline. It proved especially effective in resource-constrained environments. However, its performance was limited in some models, highlighting the importance of LLM capabilities in keyword generation and verification.

Limitations

The performance of ITERKEY is highly dependent on the capabilities of the underlying large language models (LLMs). For instance, the latest and powerful model Gemma-2, despite following instructions and adhering to the specified output format, experienced a significant drop in accuracy compared to the Baseline. This suggests that even for state-of-the-art models, achieving satisfactory performance with ITERKEY may be challenging. Consequently, the number of models capable of effectively implementing ITERKEY could be quite limited.

ITERKEY is an iterative process, and compared to non-iterative RAG (retrieval-augmented generation) methods, it introduces additional computational costs. Each step—keyword generation, document retrieval, answer generation, and verification—requires computational resources, making the method resource-intensive, especially for large-scale applications or when using complex LLMs. Moreover, ITERKEY operates with a fixed iteration limit, which may be insufficient for highly complex or ambiguous queries, potentially leading to suboptimal results.

In the answer verification stage, the model may return 'True' even when the same entity is referred to using different formats or variations of expression. In such E.g.,s, the result is evaluated using pattern matching, but due to differences in format or expression, correct answers may not be fully recognized. This can lead to an underestimation of the model's performance, as the evaluation may not accurately reflect the model's true capabilities. Consequently, some claims made in the ablation experiments may be subject to question.

Ethical Considerations

Language models are known to sometimes generate incorrect or potentially biased information. This issue is particularly significant when sensitive questions are posed to the model. While our retrieval-augmented approach is expected to mitigate this problem to some extent by grounding responses in external sources, it does not fully eliminate the risk of biased or offensive outputs. Therefore, careful consideration is required when deploying such systems in user-facing applications to avoid unintended harm. All datasets and models used in this work are publicly available under permissible licenses. The Natural Questions dataset is provided under the CC BY-SA 3.0 license, and the WebQuestions dataset is also available under the CC BY-SA 3.0 license. The data related to EntityQA are distributed according to their respective licensing terms. Additionally, the use of these datasets is permitted for research purposes. Note that in this work, we used an AI assistant tool, ChatGPT, for coding support.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Daman Arora, Anush Kini, Sayak Ray Chowdhury, Nagarajan Natarajan, Gaurav Sinha, and Amit Sharma. 2023. Gar-meets-rag paradigm for zero-shot information retrieval. *CoRR*, abs/2310.20158.

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. 2023. Task-aware retrieval with instructions. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3650–

3675, Toronto, Canada. Association for Computational Linguistics.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Yingshan Chang, Mridu Narang, Hisami Suzuki, Guihong Cao, Jianfeng Gao, and Yonatan Bisk. 2022. Webqa: Multihop and multimodal qa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16495–16504.

Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. 2023. Lift yourself up: Retrieval-augmented text generation with self-memory. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2023. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*.

Hongwei Cui, Yuyang Du, Qun Yang, Yulin Shao, and Soung Chang Liew. 2024. Llmind: Orchestrating ai and iot with llm for complex task execution. *IEEE Communications Magazine*, pages 1–7.

Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith Hall, and Ming-Wei Chang. 2023. Promptagator: Few-shot dense retrieval from 8 examples. In *The Eleventh International Conference on Learning Representations*.

Rajarshi Das, Ameya Godbole, Dilip Kavarthapu, Zhiyu Gong, Abhishek Singhal, Mo Yu, Xiaoxiao Guo, Tian Gao, Hamed Zamani, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step entity-centric information retrieval for multi-hop question answering. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 113–118, Hong Kong, China. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier

Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards

retrieval-augmented large language models. *Preprint*, arXiv:2405.06211.

Zhangyin Feng, Xiaocheng Feng, Dezhi Zhao, Maojin Yang, and Bing Qin. 2024. Retrieval-generation synergy augmented large language models. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11661–11665.

Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 981–993, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

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.

Zhicheng Guo, Sijie Cheng, Yile Wang, Peng Li, and Yang Liu. 2023. Prompt-guided retrieval augmentation for non-knowledge-intensive tasks. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10896–10912, Toronto, Canada. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2024. Atlas: few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24(1).

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *Preprint*, arXiv:2310.06825.

Zhengbao Jiang, Luyu Gao, Zhiruo Wang, Jun Araki, Haibo Ding, Jamie Callan, and Graham Neubig. 2022. Retrieval as attention: End-to-end learning of retrieval and reading within a single transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2336–2349, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 15696–15707. PMLR.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2023. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *Preprint*, arXiv:2212.14024.

Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, and Jaewoo Kang. 2023. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 996–1009, Singapore. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2024. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

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. 2020.

Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1340–1350, Florence, Italy. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Xing Han Lù. 2024. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring. *Preprint*, arXiv:2407.03618.

Rodrigo Frassetto Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *CoRR*, abs/1904.08375.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D. Manning. 2019. Answering complex open-domain questions through iterative query generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2590–2602, Hong Kong, China. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving AI tasks with chatGPT and its friends in hugging face. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8371–8384, Mexico City, Mexico. Association for Computational Linguistics.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Weiwei Sun, Hengyi Cai, Hongshen Chen, Pengjie Ren, Zhumin Chen, Maarten de Rijke, and Zhaochun Ren. 2023. Answering ambiguous questions via iterative prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7669–7683, Toronto, Canada. Association for Computational Linguistics.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy,

Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, D. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, A. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Maxim Vidgof, Stefan Bachhofner, and Jan Mendling. 2023. Large language models for business process management: Opportunities and challenges. In *Business Process Management Forum*, Lecture Notes in Business Information Processing, page 107–123, Germany. Springer.

Boshi Wang, Xiang Deng, and Huan Sun. 2022. Iteratively prompt pre-trained language models for chain of thought. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2714–2730, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Evan Wang, Federico Cassano, Catherine Wu, Yunfeng Bai, Will Song, Vaskar Nath, Ziwen Han, Sean Hendryx, Summer Yue, and Hugh Zhang. 2024a. Planning in natural language improves llm search for code generation. *Preprint*, arXiv:2409.03733.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9423, Singapore. Association for Computational Linguistics.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Wenhan Xiong, Xiang Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *International Conference on Learning Representations*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate

rather than retrieve: Large language models are strong context generators. In *The Eleventh International Conference on Learning Representations*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.

Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. 2023a. Retrieve anything to augment large language models. *Preprint*, arXiv:2310.07554.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023b. Siren's song in the ai ocean: A survey on hallucination in large language models. *Preprint*, arXiv:2309.01219.

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. In *The Twelfth International Conference on Learning Representations*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Details of experimental setting

In this study, to ensure a fair comparison of performance across multiple models, all experiments were conducted on a single NVIDIA RTX 6000 Ada GPU, with half quantization utilized for model generation. However, due to resource constraints, Llama-3.1 70B model was loaded and inferred in 4-bit mode. In this study, we use LLMs for three steps: keyword generation, answer generation, and answer verification. For each step, an appropriate Max Token Length was set, as shown in Table 5. The same settings were applied to each model for performance comparison purposes.

B Keyword Numbers Distribution in ITERKEY

Figure 4 shows the average and variance of the number of keywords generated by each model across all tasks and iterations. Most models generate between 9 and 12 keywords, with a maximum of under 30. This distribution helps analyze the relationship between keyword generation and performance in ITERKEY. Despite the size difference, both Llama-3.1-8B and Llama-3.1-70B exhibit similar trends, peaking at 8-10 keywords with nearly identical distribution shapes. On the other hand, Gemma-2 model generates the highest average number of keywords (12.39) compared to the other models, with a more concentrated distribution. However, as discussed in Section 4, 5.1 its recall rate and EM score are lower, particularly in keyword generation and iterative refinement, where it underperforms. This suggests that generating more keywords doesn't necessarily improve accuracy, likely due to issues in the verification step identified earlier. Phi-3.5-mini model shows a similar keyword generation distribution to Llama-3.1-8B, but differences in recall rates and EM scores indicate variations in keyword quality. This underscores that performance depends not just on the number of keywords but on their quality as well.

Figures 5, 6, 7, 8 and Table 7 show how keyword generation evolves over five iterations for each model. Llama-3.1-8B consistently improves by reducing average keyword count and variance, contributing to strong performance. Llama-3.1-70B maintains stable keyword generation with slightly higher averages and increasing variance, reflecting flexibility without a notable drop in quality. In contrast, Gemma-2 generates the most keywords (mean: 12.8) but shows limited refinement and low variance, possibly limiting dynamic adjustment and lowering recall and EM scores. Phi-3.5-mini starts with a high keyword count but drops sharply, with rising variance, yet shows little performance improvement, likely due to inefficient refinement.

STEP	Max Token Length
Keyword Generation	128
Answer Generation	128
Answer Verification	32

Table 5: Max token length for each step.

Model	Params	HuggingFace Name
Llama-3.1	8B	meta-llama/Llama-3.1-8B-Instruct
Llama-3.1	70B	meta-llama/Llama-3.1-70B-Instruct
Gemma-2	9B	google/Gemma-2-9b-it
Phi-3.5-mini	3.8B	microsoft/Phi-3.5-mini-instruct

Table 6: Details of the LMs for the experiments.

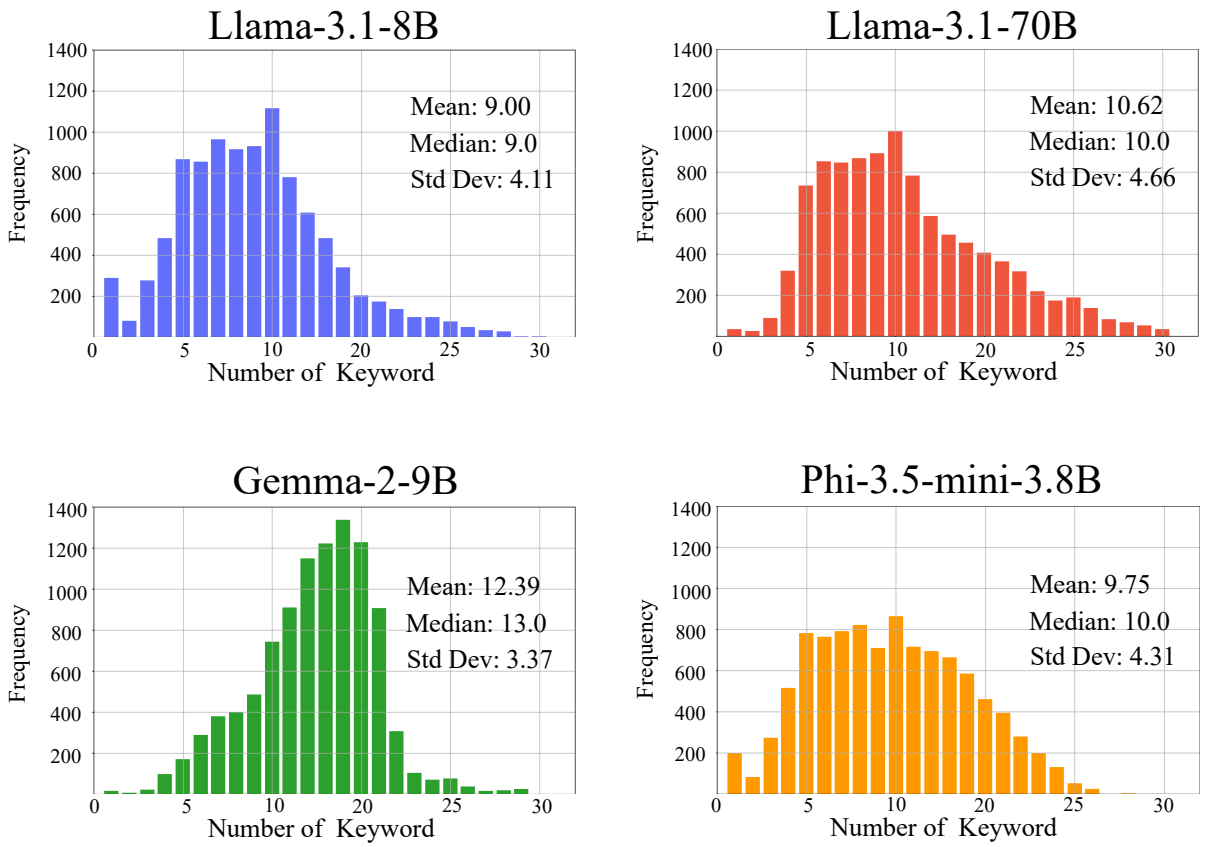

Figure 4: Destribution of keyword generated by each LLMs.

These results suggest that keyword refinement is key, with Llama models showing a strong link to performance, while Gemma-2 and Phi-3.5-mini struggle with dynamic adjustment, potentially affecting their results.

C Impact of Chunk Size on ITERKEY

This study examines the impact of different chunk sizes and Top-k retrieved documents on model performance in QA tasks. Several key trends emerged from the results, as shown in Table 8. In the 'Chunk256' 'Top3' setting, the Llama-3.1-8B model outperformed other models across all QA tasks, achieving the best performance. This indicates that retrieving Top3' documents improves the model's accuracy over Top1'. On the other hand, when only the 'Top1' document is provided, the larger Llama-3.1-70B model generally outperformed the Llama-3.1-8B model, especially in tasks like EntityQA and WebQA. This suggests that the 70B model has the capacity to leverage a smaller set of relevant information more effectively, maintaining high accuracy with fewer retrieved documents.

Furthermore, when comparing chunk sizes, the results showed that increasing the chunk size from 'Chunk256' to 'Chunk512' led to a decline in performance, particularly in the 'Top3' configuration. For example, in the 'Chunk512' 'Top3' setting, both the Llama-3.1-8B and Llama-3.1-70B models exhibited significant drops in performance across all tasks, with notable declines in EntityQA and WebQA. This suggests that larger chunk sizes exceed the model's optimal token processing capacity, causing difficulties in handling the larger input context, and consequently leading to performance degradation.

Regarding model size and information efficiency, the performance difference between Llama-3.1-8B and Llama-3.1-70B reveals an interesting tradeoff. While the 70B model benefits from a larger parameter space, allowing it to excel with fewer documents ("Top1"), the 8B model makes better use of multiple documents ('Top3'). This indicates that smaller models can compensate for their parameter limitations by utilizing more retrieved information, whereas larger models can achieve comparable or better results with minimal inputs.

Model	Iteration 1		Iteration 2		Iteration 3		Iteration 4		Iteration 5	
	Mean	Var	Mean	Var	Mean	Var	Mean	Var	Mean	Var
Llama-3.1 8B	11.2	4.1	9.3	3.9	8.5	4.0	8.1	3.8	8.1	3.9
Llama-3.1 70B	11.2	4.1	10.3	4.8	10.3	4.8	10.6	4.8	10.7	4.9
Gemma-2 9B	12.8	2.5	12.4	3.3	12.3	3.6	12.3	3.6	12.2	3.7
Phi-3.5-mini 3.8B	13.8	3.4	9.4	3.6	8.6	4.0	8.4	4.0	8.6	4.0

Table 7: Results of different models across iterations, displaying both the mean and variance.

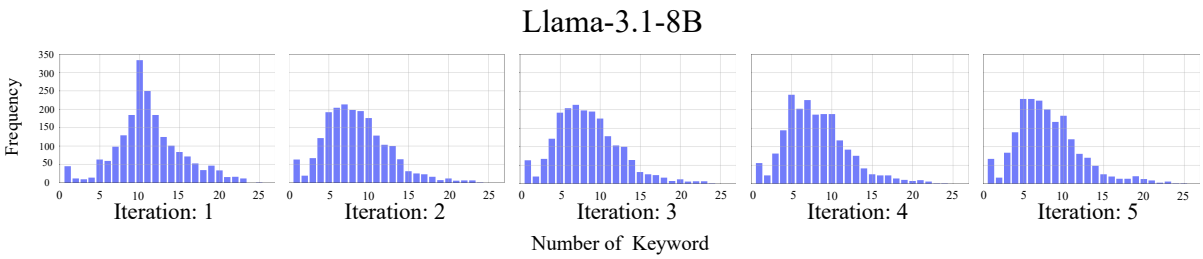

Figure 5: The distribution of the number of keywords per iteration in LLama 3.1-8B.

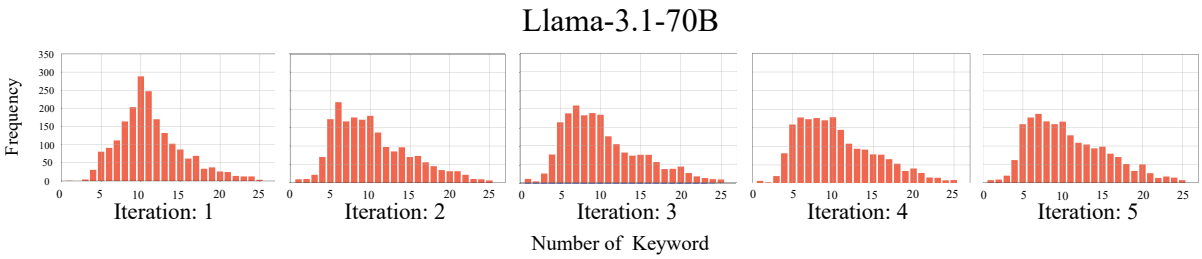

Figure 6: The distribution of the number of keywords per iteration in LLama 3.1-70B.

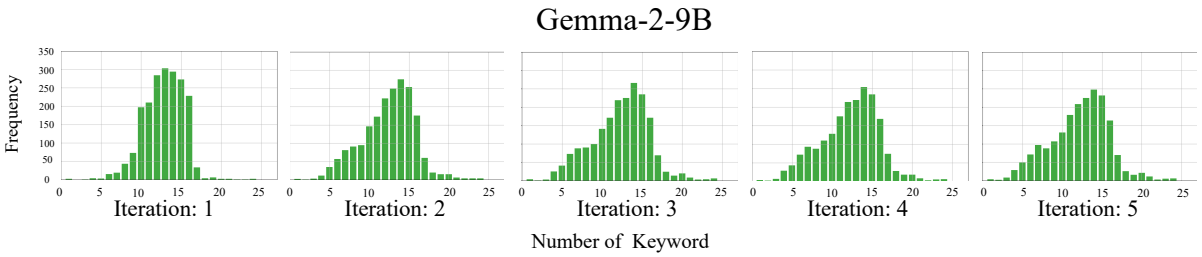

Figure 7: The distribution of the number of keywords per iteration in Gemma-2.

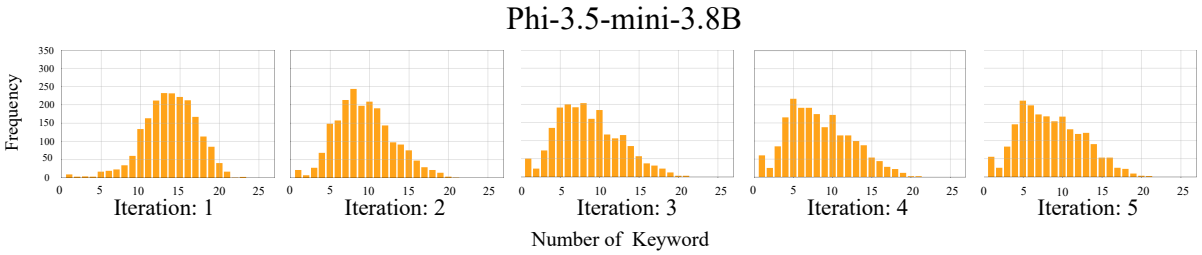

Figure 8: The distribution of the number of keywords per iteration in Phi-3.5-mini.

Method	Model	Size (B)	**Entity QA**	**HotpotQA**	**Natural QA**	**WebQA**
Chunk256 Top1	Llama-3.1	8B	46.2	41.2	47.6	56.4
	Llama-3.1	70B	53.1	44.4	49.0	55.0
	Gemma-2	9B	24.0	7.2	10.0	18.8
	Phi-3.5-mini	3.8B	34.2	28.0	28.8	43.2
Chunk256 Top3	Llama-3.1	8B	**53.6**	**47.8**	**54.8**	**59.8**
	Llama-3.1	70B	52.0	47.0	52.6	55.6
	Gemma-2	9B	7.2	7.6	8.1	13.8
	Phi-3.5-mini	3.8B	41.8	33.6	35.2	49.4
Chunk512 Top1	Llama-3.1	8B	49.6	42.2	47.2	55.2
	Llama-3.1	70B	48.7	44.4	50.4	54.3
	Gemma-2	9B	10.6	7.2	10.0	12.0
	Phi-3.5-mini	3.8B	35.2	30.0	31.9	46.1
Chunk512 Top3	Llama-3.1	8B	22.6	15.6	23.5	29.2
	Llama-3.1	70B	29.7	23.4	33.4	37.3
	Gemma-2	9B	0.6	1.6	1.0	2.1
	Phi-3.5-mini	3.8B	16.2	11.0	17.6	24.6

Table 8: Comparison of model performance on QA tasks with different chunk sizes and Top retrieved documents provided to the model during RAG. For example, with 'Chunk256' and 'Top1'

Question: Where does the PATH train stop in Newark? **Answer:** Newark Penn Station

Iteration	**Keywords**	**Model Output**	**Validation Output**	**Correct?**
1	PATH train, Newark, stops, transportation, stations, Penn Station, downtown Newark, Newark Penn Station, train stations, commute, New Jersey, NJ Transit, rail, Newark PATH stops, PATH Newark stops, Newark train stops	Answer: Newark Liberty International Airport Station	False	False
2	PATH train, Newark, stops, stations, New Jersey, NJ Transit, Penn Station, Newark Penn Station, downtown Newark, transportation, train stations, rail, commute	The PATH train stops at Newark Penn Station.	True	True

Table 9: Case 1: A case where the correct answer is reached in the second iteration.

Question: Who wrote and performed "I Can Only Imagine"? **Answer:** Bart Millard

Iteration	Keywords	Model Output	Validation Output	Correct?
1	I Can Only Imagine, MercyMe, Bart Millard, Christian Song, Christian Music Artist, Lead Vocalist, Songwriter, Christian Band, Contemporary Christian Music	Bart Millard	False	False
2	I Can Only Imagine, Bart Millard, MercyMe, Christian Song, Christian Music Artist, Lead Vocalist, Songwriter, Christian Band, Contemporary Christian Music	Bart Millard	False	False
3	I Can Only Imagine, MercyMe, Bart Millard, Christian Song, Worship Music, Christian Music Artist, Lead Vocalist, Songwriter, Christian Band	Bart Millard wrote and performed "I Can Only Imagine".	True	True

Table 10: Case 2: A case where the model returns False despite the answer being correct.

Question: What prolific English engineer, who died on 15 December 1857, contributed to the advent of the hot air engine by way of their open cycle devices? **Answer:** Sir George Cayley

Iteration	Keywords	Model Output	Validation Output	Correct?
1	English engineer, hot air engine, open cycle, December 1857	Sir George Cayley	False	False
2	English engineer, hot air engine, open cycle, 15 December 1857, inventor, hot air engine inventor	Sir George Cayley	False	False
3	English engineer, hot air engine, open cycle, inventor, 15 December 1857, British engineer, thermodynamic cycle	Sir George Cayley	False	False
4	English engineer, hot air engine, open cycle, thermodynamic cycle, 15 December 1857, British engineer, inventor, heat engine	Sir George Cayley	False	False
5	English engineer, hot air engine, open cycle, thermodynamic cycle, 1857, British engineer, inventor, heat engine, Stirling engine, Robert Stirling	Sir George Cayley	False	False

Table 11: Case 3: A case where the model fails to verify the correct answer despite arriving at it.