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

Kazuki Hayashi[†], Hidetaka Kamigaito[†], Shinya Kouda[‡], Taro Watanabe[†]

[†]Nara Institute of Science and Technology [‡]TDSE Inc.

{hayashi.kazuki.hl4, kamigaito.h, taro}@is.naist.jp

Abstract

Retrieval Augmented Generation (RAG) has emerged as a way to complement the in-context knowledge of Large Language Models (LLMs) by integrating external documents. However, real-world applications demand not only accuracy but also interpretability. Dense retrieval methods provide high accuracy but lack interpretability, while sparse retrieval is transparent but often misses query intent due to keyword matching. Thus, balancing accuracy and interpretability remains a challenge. To address these issues, we introduce ITERKEY, an LLM-driven iterative keyword generation framework that enhances RAG via sparse retrieval. ITERKEY consists of three LLM-driven stages: generating keywords for retrieval, generating answers based on retrieved documents, and validating the answers. If validation fails, the process iteratively repeats with refined keywords. Across four QA tasks, experimental results show that ITERKEY achieves 5% to 20% accuracy improvements over BM25-based RAG and simple baselines. Its performance is comparable to dense retrieval based RAG and prior iterative query refinement methods using dense models. In summary, ITERKEY is a novel BM25-based iterative RAG framework that leverages LLMs to balance accuracy and interpretability.

1 Introduction

Large Language Models (LLMs) (OpenAI, 2023; Chiang et al., 2023; Dubey et al., 2024; Abdin et al., 2024) achieve strong performance in many natural language processing tasks but still face issues like hallucinations, outdated knowledge, and complex multi-hop reasoning (Kandpal et al., 2023; Zhang et al., 2023b; Gao et al., 2024). These challenges are pronounced in knowledge-intensive tasks (Lee et al., 2019; Zellers et al., 2018), where necessary information may not be fully stored or easily retrieved within model parameters.

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 & Grave, 2021; Shuster et al., 2021; Ram et al., 2023; Izacard et al., 2024). Although improvements to individual RAG components such as query expansion, document reranking, and retrieval tuning have been extensively studied (Gao et al., 2024; Fan et al., 2024), real-world applications demand both interpretability and accuracy from the retrieval component. Dense retrieval methods are highly accurate but suffer from limited interpretability, while sparse retrieval methods are more transparent but often struggle to capture the underlying intent of user queries (Kang et al., 2025; Cheng et al., 2024; Ayoub et al., 2024; Llordes et al., 2023).

To address this, we propose ITERKEY: **Iter**ative **Key**word Generation with LLMs, a BM25-based method that improves RAG through iterative keyword refinement and self evaluation. ITERKEY operates in three stages: keyword generation, answer generation, and answer validation. In each cycle, an LLM extracts keywords, generates an answer, and checks correctness. If the validation fails, it updates the keywords and repeats the process until a satisfactory answer is obtained. By leveraging sparse BM25 retrieval, the contribution of
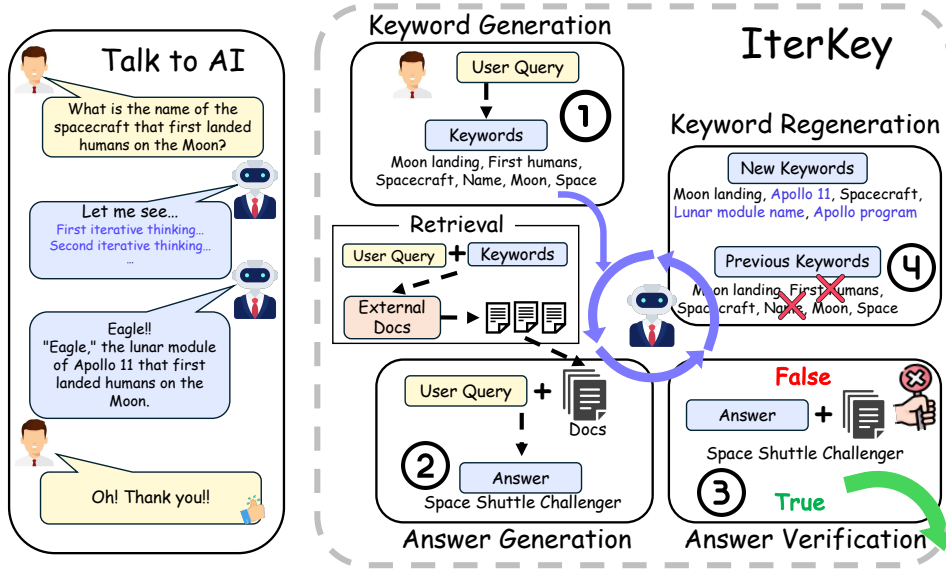
Figure 1: IterKey iteratively generates keywords, produces answers, and validates them using an LLM until the correct response is reached, thereby improving RAG accuracy.

each keyword to the score is explicit, enabling the tracking of generated keywords, retrieved documents, and validation results, which in turn improves accuracy and traceability.

Across four open domain QA datasets, ITERKEY improved retrieval performance by up to 20% over BM25 baselines and achieved 5% to 20% accuracy gains over vanilla methods and BM25-based RAG, with performance comparable to Dense model based RAG and iterative query refinement methods. These results indicate that LLMs can infer and expand keywords effectively, highlighting the practical potential of LLM-driven RAG refinement that balances performance and interpretability in supporting human search efforts.

2 ITERKEY: Iterative Keyword Generation with LLMs

Sparse retrieval algorithms, such as BM25 (Robertson & Zaragoza, 2009), efficiently handle large datasets by ranking documents based on keyword frequency and uniqueness. They are interpretable, fast, and training-free, making them suitable for scalable, low-resource settings. Their transparency lets users see which keywords influenced retrieval. However, they often miss nuanced or implicit query intent, limiting effectiveness compared to dense methods (Llordes et al., 2023; Izacard et al., 2022; Chen et al., 2023; Wang et al., 2022b). To address this limitation, we propose IterKey, a method that extends query expansion research (Wang et al., 2023a;b; Gao et al., 2023) by leveraging LLMs to iteratively refine the retrieval process in RAG. Specifically, IterKey uses the self-validation capabilities of LLMs (Wang et al., 2024) to improve sparse retrieval through iterative keyword generation, document retrieval, answer generation, and validation. By using BM25 for retrieval, all keywords that contribute to the search remain explicit, allowing the generated keywords, retrieved documents, and validation results to be inspected directly. This design aims to preserve the interpretability of sparse retrieval while improving accuracy. Table 1 summarizes the prompts used in each step of ITERKEY. We further explain the motivation behind each prompt design, with concrete examples provided in Figure 1 to illustrate the full process.

Step 1: Keyword Generation   Given a user query q, the LLM generates a set of critical keywords $\mathcal{K}^0$ essential for retrieving relevant documents. By leveraging BM25 retrieval and the query's intent, an LLM generates critical keywords from the initial query for retrieval. This step follows recent work on LLM-based query expansion (Xia et al., 2025; Lei et al., 2024) and enables the LLM to capture semantic and contextual keywords that sparse retrieval

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 } Previous Retrieval Documents: { Docs } 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: IterKey prompts for iterative keyword refinement and answer validation.

often misses, resulting in more effective document retrieval. 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*.

Step 2: Answer Generation 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, the LLM generates an answer a^i to the original query q. For example, the LLM generates the answer *Space Shuttle Challenger* using the expanded query.

Step 3: Answer Validation Recent studies (Asai et al., 2024; Yao et al., 2023) suggest that allowing an LLM to critique its own answer after generation can improve answer validity. Building on this insight, we let the LLM act as a judge to decide whether the generated answer is supported by the retrieved documents, using only the logical consistency between answer and evidence as the criterion (Zheng et al., 2023). The LLM is required to choose between 'True' or 'False' using force decoding, where the output is constrained to these two options. The generation probabilities for both are evaluated, and the option with the higher probability is selected as the answer. If 'True' is selected, the answer is considered correct, the process concludes, and a^i is returned as the final answer. If 'False' is selected, the process restarts to refine the retrieval and generation steps. After reviewing the documents, the LLM finds *Space Shuttle Challenger* incorrect and returns 'False' in our example.

Step 4: Keyword Regeneration This step follows recent work on iterative relevance feedback with LLMs (Mackie et al., 2023; Liu et al., 2024). If the LLM responds with 'False' in the validation step, we regenerate a new set of keywords $\mathcal{K}^{i+1}$ to improve retrieval. The regeneration process uses the original query and the previous keywords as cues, allowing the LLM to refine its understanding of the query intent and progressively retrieve more relevant documents. The new keywords $\mathcal{K}^{i+1}$ are then used to form a new expanded query $q + \mathcal{K}^{i+1}$, and Steps 2 through 4 are repeated. This iterative process continues until the

validation step returns 'True' or a predefined maximum number of iterations N is reached. For example, after receiving 'False', the LLM generates new keywords like *Apollo 11, Lunar module name*, and the correct answer *Eagle* is finally retrieved in Figure 1.

3   Experiments

3.1   Settings

Datasets and Evaluation Methods   We evaluated ITERKEY on four open domain QA datasets: Natural Questions (NQ) (Kwiatkowski et al., 2019), EntityQA (Li et al., 2019), We-bQA (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 per dataset due to computational cost and conducted the evaluation under zero-shot settings. We evaluate generated answers using the Exact Match (EM) metric (Rajpurkar et al., 2016), which considers an answer correct if it matches any reference after normalization (lower-casing, removing articles and punctuation, and whitespace consolidation). To assess the impact of query expansion, we compute recall as the percentage of retrieved documents that contain at least one reference answer. We used the December 2018 Wikipedia dump as the retrieval corpus for all datasets (Izacard et al., 2024) following (Feng et al., 2024).

Retrieval Models   We utilized BM25 (Robertson & Zaragoza, 2009), implemented using BM25S (Lù, 2024)[1], as the base retriever. Additionally, we adopted three Dense Models: Contriever (Izacard et al., 2022), BGE (Chen et al., 2023), and E5 (Wang et al., 2022b), under identical conditions to compare their retrieval performance against sparse models.

LLMs   To examine the practical utility of ITERKEY across different model settings, we evaluate it using four LLMs with varying capabilities: Llama-3.1 8B and 70B (Dubey et al., 2024), Gemma-2 (Team et al., 2024), and Phi-3.5-mini (Abdin et al., 2024). This analysis examines how generation and validation abilities impact each component, offering insights for practical model selection. All models can produce the structured outputs required by each ITERKEY step. See Appendices B and C for details.

Comparative Baseline   As a representative iterative baseline, we adopt ITRG (Iterative Retrieval-Generation Synergy) (Feng et al., 2024), the strongest publicly available method in this line of work. ITRG repeatedly concatenates LLM-generated answer passages with the original query to refine retrieval quality. In contrast, IterKey generates explicit keyword lists and introduces an answer validation step that halts iteration once the answer is deemed correct. ITRG integrates established single-pass query expansion techniques (Wang et al., 2023b;a; Gao et al., 2023), positioning it as an appropriate, advanced comparator.

3.2   Results

IterKey vs. Baseline & RAG (BM25)   Table 2 shows that ITERKEY consistently improves accuracy over the Baseline across all models, achieving gains of 10% to 25%. Particularly, Llama-3.1 models of 8B and 70B parameters show notable improvements with ITERKEY, with the 8B model reaching accuracy levels similar to the 70B model. This demonstrates ITERKEY's effectiveness in enhancing the performance of smaller models, allowing them to achieve accuracy levels closer to larger ones. Further comparison with BM25-based RAG reveals a 5% to 10% accuracy increase on Llama-3.1 when using ITERKEY. However, ITERKEY performs comparably to RAG (BM25) on Phi-3.5-mini, while falling behind RAG (BM25) on Gemma-2, with no notable improvements on the other tasks. These findings highlight the effectiveness of ITERKEY, although the benefits vary across models.

IterKey vs. RAG (E5) & ITRG   As shown in Table 2, ITERKEY with BM25-based retrieval outperforms E5 based RAG on Llama-3.1 models. Notably, the Llama-3.1 70B model achieves

---

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

Method	Model	Size (B)	Entity QA	HotpotQA	Natural QA	WebQA
Vanilla	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
RAG (BM25)	Llama-3.1	8B	54.0	47.0	44.8	51.4
	Llama-3.1	70B	54.6	46.2	43.4	47.4
	Gemma-2	9B	47.9	39.6	33.2	41.6
	Phi-3.5-mini	3.8B	48.2	42.2	32.6	40.2
RAG (E5)	Llama-3.1	8B	52.9	47.7	49.6	48.2
	Llama-3.1	70B	57.0	51.0	49.4	48.8
	Gemma-2	9B	52.2	40.8	41.5	41.8
	Phi-3.5-mini	3.8B	50.2	44.6	37.0	41.4
ITRG Refresh (E5)	Llama-3.1	8B	60.6	<u>53.4</u>	<u>53.6</u>	<u>56.2</u>
	Llama-3.1	70B	60.7	52.9	53.3	51.6
	Gemma-2	9B	<u>54.2</u>	<u>47.6</u>	<u>47.4</u>	<u>48.5</u>
	Phi-3.5-mini	3.8B	<u>54.3</u>	<u>47.1</u>	36.2	<u>44.6</u>
IterKey (BM25)	Llama-3.1	8B	<u>62.0</u>	49.7	50.2	53.3
	Llama-3.1	70B	**63.5**	**54.9**	**53.8**	**57.7**
	Gemma-2	9B	36.1	26.1	35.9	36.1
	Phi-3.5-mini	3.8B	47.9	42.7	36.8	41.2

Table 2: 'Vanilla' uses no retrieval. 'RAG (BM25)' and 'RAG (E5)' perform a single retrieval step on the original query using BM25 and E5, respectively. Other dense retriever results are in Table 18. 'ITRG Refresh (E5)' is a prior iterative refinement method, reproduced here with E5 (its best dense retriever) and five query expansions. Two approaches, Refine and Refresh, exist; see (Feng et al., 2024) for details. Refine results are in Table 19, while Refresh is in the main text. Our 'IterKey (BM25)' iteratively generates and refines keywords for up to five retrieval iterations to optimize answer quality. In the table, <u>underlined values</u> mark the best performance for each model, and **bold values** the highest accuracy overall.
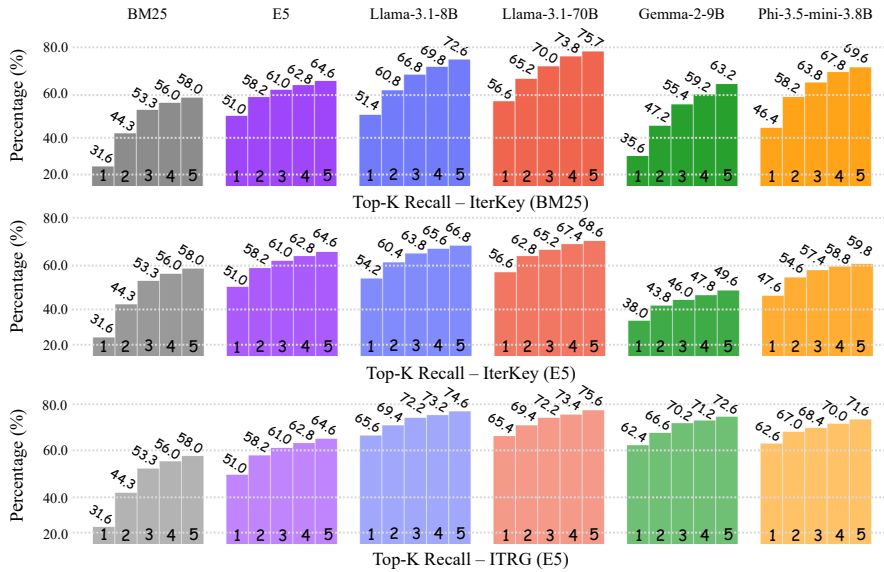

Figure 2: The figure shows the Top-K retrieval recall rates for each model in the Entity QA task, representing the proportion of retrieved documents containing the correct answer. The baseline methods, 'BM25' and 'E5,' use a single retrieval step based on the original query. Our proposed method, 'IterKey (BM25),' is compared with 'IterKey (E5)' for dense retrieval and the prior work 'ITRG (E5).' Recall rates are averaged over five iterations.

the highest accuracy across all tasks, outperforming ITRG, an iterative refinement method that employs a dense retriever. Moreover, the Llama-3.1 8B model attains accuracy comparable to ITRG, demonstrating that ITERKEY can effectively compete with dense retrieval based iterative strategies even at smaller scales. However, ITERKEY's effectiveness does not

extend uniformly across all models. While ITRG consistently improves performance for Gemma-2 and Phi-3.5-mini, ITERKEY does not achieve comparable gains in these models. Additionally, compared to E5 based RAG, its performance remains lower across most tasks.

4   Analysis

4.1   Recall and QA Performance in IterKey

Figure 2 shows recall rates (%) on the Entity QA task for three methods and four models, averaged over five iterations. For comparison, BM25 and E5 (leftmost) show a single retrieval step using the original query. Recall is the percentage of retrieved documents containing the correct answer. IterKey (BM25) improves recall by approximately 20% over standard BM25. For the Llama-3.1 models, the top-5 recall is about 10% higher than that of E5, indicating that LLMs effectively generate and expand keywords to enhance retrieval. Although Gemma and Phi outperform BM25, their recall remains below E5, implying keyword generation quality depends heavily on the model. Despite achieving approximately 60% top-3 recall, Gemma and Phi exhibit QA accuracy that is 20–30 percentage points lower (see Table 2), suggesting weaknesses in answer generation or validation significantly reduce performance.

Comparing IterKey (BM25) and IterKey (E5) reveals similar Top-1 performance; however, IterKey (BM25) achieves 10% higher Top-5 precision and higher overall QA accuracy, see Table 6. This likely reflects that plain keyword lists align well with BM25, while E5, trained for natural-language queries, handles keyword strings less effectively. These results show that our iterative keyword refinement is especially effective for sparse retrieval models.

To further analyze how accuracy evolves over iterations, Table 5 reports per-iteration results. Most performance gains occur in the first one or two iterations, with accuracy improving by up to 7%. After the third iteration, improvements taper off, indicating diminishing returns. This demonstrates that IterKey balances efficiency and effectiveness by capturing most of the benefits early while retaining the flexibility to continue when necessary.

Model	Step 2	Step 3	Step 4	Step 5	Total	Mean
Llama-3.1-8B	10.5	9.57	8.59	8.20	36.9	9.21
Llama-3.1-70B	9.34	3.65	3.78	3.26	20.0	5.01
Gemma-2	14.1	8.56	5.70	4.59	33.0	8.25
Phi-3.5-mini	18.1	7.12	6.00	5.07	36.3	9.07

Model	Step 2	Step 3	Step 4	Step 5	Total	Mean
Llama-3.1-8B	2.12	1.65	1.53	1.32	6.62	1.66
Llama-3.1-70B	1.93	1.44	1.15	1.25	5.77	1.44
Gemma-2	2.22	1.61	1.27	1.12	6.22	1.55
Phi-3.5-mini	2.03	0.69	1.04	0.95	4.71	1.18

Table 3: Average number of new keywords generated per query at each iteration.

Table 4: Average number of new documents retrieved per query at each iteration.

4.2   Keyword and Document Expansion by Iteration

To clarify the role of each iteration in our method, we show the average number of new keywords and documents added at each step. Table 3 presents the average number of newly generated keywords per query at each iteration from Step 2 to Step 5. Table 4 shows the corresponding statistics for newly retrieved documents. These results show that each iteration continues to introduce new information. On average, each step generates between 3 and 18 additional keywords per query, resulting in a total of 20 to 37 new keywords after four steps. Similarly, each iteration retrieves approximately 0.7 to 2.2 new documents per query, with a total of 4.7 to 6.6 documents accumulated through the full process. These findings indicate that later iterations are effective in generating novel keywords and retrieving previously unseen documents. The iterative process enables the method to maintain both efficiency and coverage, as it allows early termination for easier queries while providing additional refinement for more challenging cases.

4.3   Keyword Generation Step

Based on these results, we conducted additional experiments to further analyze the impact of keyword quality on ITERKEY's performance. Specifically, we evaluated the effects of

Model	Iter 1	Iter 2		Iter 3		Iter 4		Iter 5	
Llama-3.1 8B	62.2	65.4	+3.2	69.6	+7.4	70.5	+8.3	72.6	+10.4
Llama-3.1 70B	65.7	70.2	+4.5	73.5	+7.8	74.4	+8.7	75.7	+10.0
Gemma-2 9B	53.2	57.6	+4.4	60.0	+6.8	61.2	+8.0	63.2	+10.0
Phi-3.5 mini	57.5	64.5	+7.0	67.9	+10.4	68.8	+11.3	69.6	+12.1

Table 5: Accuracy (%) on the Entity QA task for each iteration. Values in blue boxes show absolute gains from Iter 1.

Method	Model	Size (B)	Entity QA		HotpotQA		Natural QA		WebQA	
IterKey (BM25)	Llama-3.1	8B		62.0		49.7		50.2		53.3
	Llama-3.1	70B		63.5		54.9		53.8		57.7
	Gemma-2	9B		36.1		26.1		35.9		36.1
	Phi-3.5-mini	3.8B		47.9		42.7		36.8		41.2
IterKey (E5)	Llama-3.1	8B	-1.8	60.2	+4.1	53.8	+1.2	51.4	-2.7	50.6
	Llama-3.1	70B	-1.3	62.2	-2.1	52.8	-0.6	53.2	-5.7	52.0
	Gemma-2	9B	-1.6	34.5	-0.9	25.2	-0.7	35.2	-3.7	32.4
	Phi-3.5-mini	3.8B	-0.5	47.4	+1.8	44.5	-2.4	34.4	-1.0	40.2

Table 6: Performance of ITERKEY with 'BM25' and the Dense Model ('E5'). This table shows the QA performance when replacing 'BM25' with 'E5' under consistent settings.

using high quality keywords generated by Llama 3.1 70B, which achieved the highest recall, and low quality keywords from Gemma 2, which had the lowest recall. As shown in Table 7, using high-quality keywords consistently improved accuracy across all models, notably resulting in an about 5% gain for Gemma 2. Conversely, employing low-quality keywords degraded performance for all models and tasks, with Llama 3.1 (8B) experiencing a particularly significant accuracy drop of 4.5% on EntityQA. These results clearly demonstrate that keyword quality plays a crucial role in ITERKEY's performance.

Notably, we observed that even when provided with high-quality keywords, Gemma-2's accuracy still lagged considerably behind Llama-3.1. Given that ITERKEY iteratively refines retrieval and validation, this suggests that Gemma-2's primary performance bottleneck lies in its validation capability rather than in keyword generation.

4.4 Answer Validation Step

We conduct additional experiments to examine validation model quality affects IterKey performance. We compare the high quality validation model Llama 3.1 70B, which achieves the highest baseline accuracy, with the low quality validation model Gemma, which shows the lowest baseline accuracy. As shown in Table 5, using a high quality validation model consistently improves accuracy across all base models. Gemma improves by more than five points on average over four tasks. In contrast, Gemma lowers accuracy for all models, with LLaMA-based models dropping by over ten points, showing validation quality is critical for IterKey and that Gemma's limited validation capability severely constrains its performance.

We also examined how IterKey's Answer Validation step performs over repeated iterations. Specifically, we iterated up to five times, including after a 'True' judgment, and compared accuracy across three settings to assess each model's validation performance. As shown in Table 9, comparing VerifiedAll' with Base' reveals substantial discrepancies in validation accuracy across all models, with Gemma-2 showing the largest gaps (16.9% in HotpotQA and 15.2% in Entity QA). These results indicate that validation errors are more prominent in weaker models, highlighting the challenge of ensuring reliable answer verification.

The miss True rate ('Base' vs. 'VerifiedTrue') measures the tendency to accept wrong answers as correct, while the miss False rate ('VerifiedTrue' vs. 'VerifiedAll') quantifies failure to recognize correct answers under stricter validation. For Llama-3.1-8B, Llama-3.1-70B, and Phi-3.5-mini, 'miss True' is consistently higher than 'miss False,' indicating a tendency to

Method	Model	Size (B)	Entity QA		HotpotQA		Natural QA		WebQA	
IterKey	Llama-3.1	8B		62.0		49.7		50.2		53.3
	Llama-3.1	70B		63.5		54.9		53.8		57.7
	Gemma-2	9B		36.1		26.1		35.9		36.1
	Phi-3.5-mini	3.8B		47.9		42.7		36.8		41.2
IterKey **w/ HQ Keywords**	Llama-3.1	8B	+1.4	63.4	+3.1	52.8	+1.4	51.6	+0.7	54.0
	Llama-3.1	70B		63.5		54.9		53.8		57.7
	Gemma-2	9B	+3.9	40.0	+4.4	30.5	+1.6	37.5	+3.9	40.0
	Phi-3.5-mini	3.8B	+1.9	49.8	+1.5	44.2	+1.2	38.0	+2.2	43.4
IterKey **w/ LQ Keywords**	Llama-3.1	8B	-6.0	56.0	-0.5	49.2		50.2	-1.4	51.9
	Llama-3.1	70B	-6.4	57.1	-2.6	52.3	-3.3	50.5	-5.1	52.6
	Gemma-2	9B		36.1		26.1		35.9		36.1
	Phi-3.5-mini	3.8B	-2.8	45.1	-2.2	40.5	-2.4	34.4	+0.2	41.4

Table 7: Ablation study on keyword quality in IterKey. 'High-quality (HQ) keywords' are generated by Llama-3.1-70B, the model with the highest retrieval recall, and reused by smaller models with up to 10× fewer parameters, consistently boosting performance. 'Low-quality (LQ) keywords,' generated by Gemma-2-9B, degrade performance across models.

Method	Model	Size (B)	Entity QA		HotpotQA		Natural QA		WebQA	
IterKey	Llama-3.1	8B		62.0		49.7		50.2		53.3
	Llama-3.1	70B		63.5		54.9		53.8		57.7
	Gemma-2	9B		36.1		26.1		35.9		36.1
	Phi-3.5-mini	3.8B		47.9		42.7		36.8		41.2
IterKey w/ **HQ Validation Model**	Llama-3.1	8B	+0.5	62.5	+2.0	51.7	+2.0	52.2	+1.2	54.5
	Llama-3.1	70B		63.5		54.9		53.8		57.7
	Gemma-2	9B	+7.4	43.5	+8.0	34.1	+4.6	40.5	+3.3	39.4
	Phi-3.5-mini	3.8B	+2.9	50.8	+1.9	44.6	+1.0	37.8	+2.2	43.4
IterKey w/ **LQ Validation Model**	Llama-3.1	8B	-10.4	51.6	-8.0	41.7	-5.7	44.5	-7.1	46.2
	Llama-3.1	70B	-10.7	52.8	-11.9	43.0	-11.4	42.4	-5.9	51.8
	Gemma-2	9B		36.1		26.1		35.9		36.1
	Phi-3.5-mini	3.8B	-2.6	45.3	-2.6	40.1	-4.0	32.8	-3.0	38.2

Table 8: Performance comparison of the IterKey validation step. The 'High Quality' uses Llama-3.1-70B for the validation step, whereas the 'Low Quality' employs Gemma, the model with the lowest baseline accuracy. Only the validation step is modified, and the performance changes across four tasks (improvements in blue, degradations in red) are shown.

accept incorrect answers and limiting accuracy gains. In contrast, Gemma-2 shows a higher 'miss False' rate, reflecting difficulty identifying correct answers under stricter validation.

Overall, these results show that validation errors, particularly 'miss True,' significantly constrain accuracy improvements. Thus, a model's validation capability is essential when applying IterKey. Moreover, adherence to output format does not necessarily imply strong instruction comprehension. Even the models that follow formatting constraints may fail to apply validation rules accurately, as seen in Gemma-2's validation errors. This aligns with prior studies (Kung & Peng, 2023; Zhou et al., 2023b; Liang et al., 2024) emphasizing the distinction between format adherence and instruction comprehension in LLMs.

4.5 Keyword Regeneration Prompt

We compared IterKey (BM25), which reuses previous keywords as a proxy for documents in subsequent iterations, with a document-by-document approach that regenerates and concatenates keywords for each retrieved document for the next retrieval. Since we use three documents at retrieval time, in a single iteration, the model ends up generating keywords two additional times. The prompt we used is shown in Table 10.

As shown in Table 11, the approach that generates keywords based on each retrieved document achieves higher recall for Gemma-2. However, for Llama-3.1 (8B, 70B) and Phi-3.5-mini, our original IterKey (BM25) approach still produces higher accuracy. Moreover, from a cost performance perspective, using previously generated keywords as a proxy for

Setting	Model	Size (B)	Entity QA		HotpotQA		Natural QA		WebQA	
Base	Llama-3.1	8B		62.0		49.7		50.2		53.3
	Llama-3.1	70B		63.5		54.9		53.8		57.7
	Gemma-2	9B		36.1		26.1		35.9		36.1
	Phi-3.5-mini	3.8B		47.9		42.7		36.8		41.2
VerifiedTrue	Llama-3.1	8B	+1.9	63.9	+5.3	55.0	+4.3	54.5	+4.5	57.8
	Llama-3.1	70B	+2.3	65.8	+2.1	57.0	+4.0	57.8	+4.7	62.4
	Gemma-2	9B	+4.9	41.0	+8.1	34.2	+3.5	39.4	+4.4	40.5
	Phi-3.5-mini	3.8B	+4.6	52.5	+4.4	47.1	+3.7	40.5	+5.6	46.8
VerifiedAll	Llama-3.1	8B	+4.6	66.6	+8.9	58.6	+10.0	60.2	+7.5	60.8
	Llama-3.1	70B	+4.3	67.8	+6.6	61.5	+10.2	64.0	+6.5	64.2
	Gemma-2	9B	+14.3	50.4	+16.4	42.5	+7.9	43.8	+11.5	47.6
	Phi-3.5-mini	3.8B	+7.7	55.6	+7.5	50.2	+8.6	45.4	+11.2	52.4

Table 9: Accuracy results for different models and settings across four QA tasks. **Settings**: 'Base' uses the first "True" in the Answer Verification step as the answer, then stops. 'VerifiedTrue' checks if any iteration judged "True" contains the correct answer. 'VerifiedAll' checks all iterations, both "True" and "False," for the correct answer.

Step	Prompt
Step 4: Keyword Regeneration with Docs	**System:** Refine the provided keywords to enhance document retrieval accuracy for BM25 search in the RAG framework. **User:** Please refine the keyword selection process to improve the accuracy of retrieving documents containing the correct answer. **Query:** { q } Previous Keywords: { K } Previous Retrieval Documents: { Docs } Provide the refined list of keywords in this format: ["keyword1", "keyword2", ...]. Ensure each keyword is separated by a comma, and do not include any additional text.

Table 10: Compared to the Step 4 prompt in Table 1, this prompt incrementally adds 'Previous Retrieval Documents' at a time and regenerates new keywords at each step.

documents remains more efficient. Hence, while the document-by-document approach is effective for Gemma-2, our method of using previously generated keywords as a proxy offers a more favorable balance of accuracy and cost in most cases.

4.6 Computational Cost

We evaluated how well ITERKEY balances accuracy and computational cost compared to other methods. Table 12 compares runtime for the Entity QA dataset using Llama-3.1-8B. For a fair comparison, both ITERKEY and ITRG used E5 for retrieval. Inference was performed on an NVIDIA RTX 6000 Ada GPU, while BM25 retrieval used an Intel Xeon Platinum 8160 CPU. As a result, IterKey achieved comparable accuracy to ITRG while using only approximately 80% of the latency per question, with an average latency of 3.43 seconds.

Table 13 shows the average iterations required by ITERKEY across all QA tasks and models. The count stays below 1.5, with each iteration involving three LLM calls: keyword generation, answer generation, and validation. Multiplying the 3.43 s base latency by the average 1.5 iterations yields ≈ 5.15 s per question. In summary, while iterative reasoning introduces a certain overhead (about 4 seconds slower than non-iterative baselines), the per-question latency remains well within acceptable bounds for real-time QA with GPU-based inference. Given the substantial improvements in accuracy and robustness, we consider this a reasonable trade-off in practical settings where reliability and interpretability are essential.

5 Related Work

Retrieval Augmented Generation (RAG) RAG enhances response accuracy and relevance by incorporating external knowledge, making it effective for open-domain QA (Lewis et al., 2020; Izacard & Grave, 2021; Shuster et al., 2021; 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).

Method	Model	Size (B)	Top1 (%)	Top2 (%)	Top3 (%)	Top4 (%)	Top5 (%)
IterKey (BM25)	Llama-3.1	8B	51.4	60.8	66.8	69.8	72.6
	Llama-3.1	70B	56.6	65.2	70.0	73.8	75.7
	Gemma-2	9B	35.6	47.2	55.4	59.2	63.2
	Phi-3.5-mini	3.8B	46.4	58.2	63.8	67.8	69.6
IterKey (BM25) document-by-document	Llama-3.1	8B	+3.6 55.0	-0.8 60.0	-3.4 63.4	-4.6 65.2	-4.6 68.0
	Llama-3.1	70B	+2.4 59.0	+0.2 65.4	-1.8 68.2	-3.4 70.4	-2.9 72.8
	Gemma-2	9B	+15.4 51.0	+8.8 56.0	+2.8 58.2	+3.0 62.2	+2.2 65.4
	Phi-3.5-mini	3.8B	+2.4 48.8	-4.2 54.0	-8.0 55.8	-7.2 60.6	-6.4 63.2

Table 11: Retrieval performance comparison between the original IterKey (BM25) and the document-by-document keyword regeneration approach.

Step	RAG(BM25)	RAG(E5)	ITRG	IterKey
Query Expansion	–	–	–	1.33
Retrieval	0.44	0.11	0.88	0.33
Answer Generation	0.64	0.63	3.39	1.05
Answer Validation	–	–	–	0.72
Total	1.08	0.73	4.26	3.43

Model	Size (B)	Entity	Hotpot	Natural	Web
Llama-3.1	8B	1.27	1.38	1.35	1.36
Llama-3.1	70B	1.15	1.20	1.23	1.10
Gemma-2	9B	1.33	1.36	1.34	1.37
Phi-3.5-mini	3.8B	1.38	1.32	1.28	1.29

Table 12: Average latency (s) for each step in Entity QA with LLaMA-3.1-8B.

Table 13: Average iterations for ITERKEY.

Recent work has focused on improving components such as query expansion, retrieval tuning, 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), yet these processes often function independently, making coherence difficult (Shi et al., 2024; Guo et al., 2023). Even when relevant documents are retrieved, they may not be fully used in generating responses (Ram et al., 2023; Dai et al., 2023; Xu et al., 2024; Asai et al., 2024), and retrieval failures can lead to irrelevant outputs (Jiang et al., 2023b; Guo et al., 2023).

Query and Document Expansion   Recent methods improve retrieval by generating pseudo documents from queries, benefiting both sparse and dense systems (Wang et al., 2023a). Generating hypothetical documents has also enhanced zero-shot dense retrieval without relevance labels (Gao et al., 2023). These approaches leverage LLMs to address short or ambiguous queries by adding relevant context (Zhang et al., 2023b; Asai et al., 2023).

Iterative and Chain-of-Thought Retrieval   Iterative retrieval methods leverage intermediate reasoning to refine queries for complex, multi-step information needs, integrating generative and Chain-of-Thought (CoT) reasoning to dynamically update queries and responses (Shao et al., 2023; Kim et al., 2023; Feng et al., 2024; Sun et al., 2023; Jiang et al., 2023b; Arora et al., 2023; Wang et al., 2022a). CoT-guided retrieval further enhances document relevance, factual accuracy, and reduces hallucinations (Trivedi et al., 2023; Creswell et al., 2023; Zhou et al., 2023a; Yao et al., 2023; Wei et al., 2022; Kojima et al., 2024; Zheng et al., 2024). Both supervised and unsupervised iterative approaches excel in multi-hop QA and knowledge-intensive tasks, making them effective even in few-shot and zero-shot scenarios (Das et al., 2019; Xiong et al., 2021; Khattab et al., 2023).

6   Conclusion

We introduce ITERKEY, a novel BM25-based framework that refines Retrieval Augmented Generation (RAG) by dividing the process into three stages: keyword generation, answer generation, and answer validation, all performed iteratively by an LLM. Experimental results across four open domain QA datasets show that ITERKEY improves retrieval performance by 20% over standard BM25 baselines and achieves 5% to 20% accuracy gains over vanilla and BM25-based RAG. It also delivers performance comparable to Dense model-based RAG and prior iterative refinement methods. These results demonstrate that LLMs can infer and refine keywords effectively to improve retrieval and answer quality, while preserving the transparency of sparse retrieval. Our findings highlight the potential of LLMs to enhance both accuracy and interpretability in real-world RAG applications.

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. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

Daman Arora, Anush Kini, Sayak Ray Chowdhury, Nagarajan Natarajan, Gaurav Sinha, and Amit Sharma. Gar-meets-rag paradigm for zero-shot information retrieval. *CoRR*, abs/2310.20158, 2023. URL http://dblp.uni-trier.de/db/journals/corr/corr2310.html#abs-2310-20158.

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. Task-aware retrieval with instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 3650–3675, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.225. URL https://aclanthology.org/2023.findings-acl.225.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Michael Antonios Kruse Ayoub, Zhan Su, and Qiuchi Li. A case study of enhancing sparse retrieval using llms. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24,

pp. 1609–1615, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701726. doi: 10.1145/3589335.3651945. URL https://doi.org/10.1145/3589335.3651945.

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. Qwen technical report, 2023. URL https://arxiv.org/abs/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. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

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

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2023.

Yiruo Cheng, Kelong Mao, and Zhicheng Dou. Interpreting conversational dense retrieval by rewriting-enhanced inversion of session embedding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2879–2893, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.159. URL https://aclanthology.org/2024.acl-long.159/.

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. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3Pf3Wg6o-A4.

Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith Hall, and Ming-Wei Chang. Promptagator: Few-shot dense retrieval from 8 examples. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=gmL46YMpu2J.

Rajarshi Das, Ameya Godbole, Dilip Kavarthapu, Zhiyu Gong, Abhishek Singhal, Mo Yu, Xiaoxiao Guo, Tian Gao, Hamed Zamani, Manzil Zaheer, and Andrew McCallum. Multi-step entity-centric information retrieval for multi-hop question answering. In Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen (eds.), *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pp. 113–118, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5816. URL https://aclanthology.org/D19-5816.

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. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models, 2024. URL https://arxiv.org/abs/2405.06211.

Zhangyin Feng, Xiaocheng Feng, Dezhi Zhao, Maojin Yang, and Bing Qin. Retrieval-generation synergy augmented large language models. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 11661–11665, 2024. doi: 10.1109/ICASSP48485.2024.10448015.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1762–1777, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.99. URL https://aclanthology.org/2023.acl-long.99.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024. URL https://arxiv.org/abs/2312.10997.

Zhicheng Guo, Sijie Cheng, Yile Wang, Peng Li, and Yang Liu. Prompt-guided retrieval augmentation for non-knowledge-intensive tasks. In Anna Rogers, Jordan Boyd-Graber,

and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10896–10912, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.693. URL https://aclanthology.org/2023.findings-acl.693.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 874–880, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.74. URL https://aclanthology.org/2021.eacl-main.74.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning, 2022. URL https://arxiv.org/abs/2112.09118.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24(1), March 2024. ISSN 1532-4435.

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. Mistral 7b, 2023a. URL https://arxiv.org/abs/2310.06825.

Zhengbao Jiang, Luyu Gao, Zhiruo Wang, Jun Araki, Haibo Ding, Jamie Callan, and Graham Neubig. Retrieval as attention: End-to-end learning of retrieval and reading within a single transformer. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2336–2349, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.149. URL https://aclanthology.org/2022.emnlp-main.149.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, Singapore, December 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 15696–15707. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kandpal23a.html.

Hao Kang, Tevin Wang, and Chenyan Xiong. Interpret and control dense retrieval with sparse latent features, 2025. URL https://arxiv.org/abs/2411.00786.

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp, 2023. URL https://arxiv.org/abs/2212.14024.

Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, and Jaewoo Kang. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 996–1009, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.63. URL https://aclanthology.org/2023.emnlp-main.63.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 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, 2024. Curran Associates Inc. ISBN 9781713871088.

Po-Nien Kung and Nanyun Peng. Do models really learn to follow instructions? an empirical study of instruction tuning. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1317–1328, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.113. URL https://aclanthology.org/2023.acl-short.113/.

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. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6086–6096, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1612. URL https://aclanthology.org/P19-1612.

Yibin Lei, Yu Cao, Tianyi Zhou, Tao Shen, and Andrew Yates. Corpus-steered query expansion with large language models. In Yvette Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 393–401, St. Julian's, Malta, March 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.eacl-short.34/.

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. 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, 2020. Curran Associates Inc. ISBN 9781713829546.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. Entity-relation extraction as multi-turn question answering. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1340–1350, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1129. URL https://aclanthology.org/P19-1129.

Shihao Liang, Runchu Tian, Kunlun Zhu, Yujia Qin, Huadong Wang, Xin Cong, Zhiyuan Liu, Xiaojiang Liu, and Maosong Sun. Exploring format consistency for instruction tuning, 2024. URL https://arxiv.org/abs/2307.15504.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In Eneko Agirre, Marianna Apidianaki, and Ivan Vulić (eds.), *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pp. 100–114, Dublin, Ireland and Online, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.deelio-1.10. URL https://aclanthology.org/2022.deelio-1.10.

Yanming Liu, Xinyue Peng, Xuhong Zhang, Weihao Liu, Jianwei Yin, Jiannan Cao, and Tianyu Du. RA-ISF: Learning to answer and understand from retrieval augmentation via iterative self-feedback. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4730–4749, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.281. URL https://aclanthology.org/2024.findings-acl.281/.

Michael Llordes, Debasis Ganguly, Sumit Bhatia, and Chirag Agarwal. Explain like i am bm25: Interpreting a dense model's ranked-list with a sparse approximation, 2023. URL https://arxiv.org/abs/2304.12631.

Xing Han Lù. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring, 2024. URL https://arxiv.org/abs/2407.03618.

Iain Mackie, Shubham Chatterjee, and Jeffrey Dalton. Generative relevance feedback with large language models. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, pp. 2026–2031. ACM, July 2023. doi: 10.1145/3539618.3591992. URL http://dx.doi.org/10.1145/3539618.3591992.

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. URL https://arxiv.org/abs/2303.08774.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL https://aclanthology.org/D19-1250.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331, 2023. doi: 10.1162/tacl_a_00605. URL https://aclanthology.org/2023.tacl-1.75.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5418–5426, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.437. URL https://aclanthology.org/2020.emnlp-main.437.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389, 01 2009. doi: 10.1561/1500000019.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9248–9274, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.620. URL https://aclanthology.org/2023.findings-emnlp.620.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: Retrieval-augmented black-box language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.463. URL https://aclanthology.org/2024.naacl-long.463.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. In Marie-Francine Moens, Xuanjing Huang,

Lucia Specia, and Scott Wen-tau Yih (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 3784–3803, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.320. URL https://aclanthology.org/2021.findings-emnlp.320.

Weiwei Sun, Hengyi Cai, Hongshen Chen, Pengjie Ren, Zhumin Chen, Maarten de Rijke, and Zhaochun Ren. Answering ambiguous questions via iterative prompting. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7669–7683, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.424. URL https://aclanthology.org/2023.acl-long.424.

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. Gemma: Open models based on gemini research and technology, 2024. URL https://arxiv.org/abs/2403.08295.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem 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, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557.

Boshi Wang, Xiang Deng, and Huan Sun. Iteratively prompt pre-trained language models for chain of thought. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2714–2730,

Abu Dhabi, United Arab Emirates, December 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.174. URL https://aclanthology.org/2022.emnlp-main.174.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022b.

Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query expansion with large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9414–9423, Singapore, December 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.585. URL https://aclanthology.org/2023.emnlp-main.585.

Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query expansion with large language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023b. URL https://openreview.net/forum?id=QH4EMvwF8I.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Yu Xia, Junda Wu, Sungchul Kim, Tong Yu, Ryan A. Rossi, Haoliang Wang, and Julian McAuley. Knowledge-aware query expansion with large language models for textual and relational retrieval. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4275–4286, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.216. URL https://aclanthology.org/2025.naacl-long.216/.

Wenhan Xiong, Xiang Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. Answering complex open-domain questions with multi-hop dense retrieval. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=EMHoBG0avc1.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mlJLVigNHp.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259.

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

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 93–104, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1009. URL https://aclanthology.org/D18-1009.

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

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. Siren's song in the ai ocean: A survey on hallucination in large language models, 2023b. URL https://arxiv.org/abs/2309.01219.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3bq3jsvcQ1.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

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. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=WZH7099tgfM.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023b. URL https://arxiv.org/abs/2311.07911.

A   Limitations

The performance of ITERKEY is highly dependent on the capabilities of the underlying large language model (LLM). For example, Gemma-2, a recent high-performance model, showed a notable drop in accuracy despite adhering to the output format and instructions. This suggests that even advanced models may struggle to effectively perform keyword generation or answer validation, limiting the range of LLMs suitable for ITERKEY. This aligns with prior studies (Kung & Peng, 2023; Zhou et al., 2023b; Liang et al., 2024) emphasizing the distinction between format adherence and instruction comprehension in LLMs.

ITERKEY introduces additional computational cost compared to non-iterative RAG methods. Each iteration involves keyword generation, retrieval, answer generation, and validation, resulting in higher runtime and resource usage, especially with large models or in large-scale applications. Additionally, since ITERKEY operates under a fixed iteration limit, it may fail to resolve highly ambiguous or complex queries within the allotted steps.

Finally, the answer validation step relies on textual pattern matching, which can misjudge semantically correct answers expressed in different formats such as abbreviations or paraphrases. As a result, true positives may be overlooked, leading to underestimation of model performance. This limitation particularly affects the accuracy of the ablation analysis, where validation quality is a central factor.

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, the Llama-3.1 70B model was loaded and inferred in 4-bit mode. We use LLMs for three steps: keyword generation, answer generation, and answer validation. For each step, an appropriate Max Token Length was set, as shown in Table 14. The same settings were applied to each model for performance comparison purposes.

STEP	Max Token Length
Keyword Generation	50
Answer Generation	50
Answer Validation	30

Table 14: Max token length for each step.

Table 15 provides details of the language models used in our experiments, including their parameter sizes and Hugging Face model identifiers. These models were selected based on their performance and availability for fair evaluation across different LLM architectures.

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 15: Details of the LMs for the experiments.

B Details of Experimental Setting

C Selection of Models for Iterkey

For the selection of models used in Iterkey, we conducted preliminary experiments with multiple candidate models. Based on the results, we selected models according to two main criteria (Section 2) required to implement our method without human intervention.

1. **Adherence to Specified Output Format:** The model must follow the designated generation format. In keyword generation and validation, 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 validation and keyword regeneration, it should produce consistent outputs aligned with the task requirements.

Models such as Llama 2 (Touvron et al., 2023), Qwen 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.

D Effect of Chain of Thought Reasoning

To examine whether explicit Chain of Thought (CoT) reasoning can further boost ITERKEY, we replaced the original single sentence prompts in Answer Validation (Step 3) and Keyword Regeneration (Step 4) with step by step versions. Specifically, in Step 3 the model first writes out its reasoning, citing the retrieved documents, and only then outputs 'True' or 'False'; in Step 4 it diagnoses why the previous keywords were insufficient before proposing a refined comma separated list. These changes preserve the BM25 backbone and add no extra retrieval overhead. Table 16 lists the modified prompts.

Table 17 compares the original ITERKEY (BM25) with its CoT variant on four QA benchmarks. Across models and tasks, the CoT variant fails to produce a consistent trend: NaturalQA

Step	Prompt (CoT version)
Step 3: Answer Validation (CoT)	**System:** You are an assistant that verifies whether an answer is correct based on retrieved documents. **User:** Question: { q } Answer: { a } Documents: { $Docs$ } Let's think step by step. **Chain of thought:** (Explain your reasoning step by step. Refer to the documents when relevant.) **Conclusion:** True **or** False *Do not output anything besides the above format.*
Step 4: Keyword Regeneration (CoT)	**System:** You refine keywords to improve document retrieval for BM25 search in the RAG framework. **User:** The previous keywords failed to retrieve useful documents for the query: { q } Here are the previous keywords: { *prev_keywords* } Let's think step by step. Check if they are too general, too specific, or missing important concepts. Then, refine them to produce a more effective list of keywords for retrieving documents that contain the correct answer. Output the keywords in the exact format: ["kw1", "kw2", ...] Refined Keywords:

Table 16: CoT enhanced prompts used in Step 3 (Answer Validation) and Step 4 (Keyword Regeneration).

Setting	Model	Size (B)	Entity QA	HotpotQA	Natural QA	WebQA
IterKey (BM25)	Llama-3.1	8B	62.0	49.7	50.2	53.3
	Llama-3.1	70B	63.5	54.9	53.8	57.7
	Gemma-2	9B	36.1	26.1	35.9	36.1
	Phi-3.5-mini	3.8B	47.9	42.7	36.8	41.2
IterKey (BM25) + CoT	Llama-3.1	8B	-1.6 60.4	+1.1 50.8	+3.0 53.2	-1.1 52.2
	Llama-3.1	70B	-0.3 63.2	+0.5 55.4	+1.6 55.4	-2.1 55.6
	Gemma-2	9B	-3.7 32.4	-1.5 24.6	-5.9 30.0	-5.1 31.0
	Phi-3.5-mini	3.8B	+0.3 48.2	-0.3 42.4	-0.2 36.6	-0.6 40.6

Table 17: Score changes when applying CoT to IterKey (blue indicates improvement, red indicates decline).

improves modestly by up to +1.6, whereas EntityQA, HotpotQA, and WebQA show declines of up to -2.8. These results indicate that the clearly defined instructions in each step of ITERKEY are sufficient to leverage the LLM's capabilities, making additional CoT prompting unnecessary in this BM25 centric setting.

E Keyword Numbers Distribution in ITERKEY

Figure 3 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 3.2, 4.1 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 validation 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 4, 5, 6, 7 and Table 20 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. 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.

F   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 21. In these additional experiments, we used the December 2018 Wikipedia dump as the retrieval corpus for all datasets (Izacard et al., 2024). The text was divided into segments of tokens, with an overlap of 50 tokens at the beginning and end of each chunk to preserve context.

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.

Method	Model	Size (B)	Entity QA	HotpotQA	Natural QA	WebQA
Vanilla	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
RAG (BM25)	Llama-3.1	8B	54.0	47.0	44.8	51.4
	Llama-3.1	70B	54.6	46.2	43.4	47.4
	Gemma-2	9B	47.9	39.6	33.2	41.6
	Phi-3.5-mini	3.8B	48.2	42.2	32.6	40.2
RAG (E5)	Llama-3.1	8B	50.9	46.7	47.2	48.2
	Llama-3.1	70B	56.5	52.0	46.4	45.0
	Gemma-2	9B	52.2	40.8	41.5	41.8
	Phi-3.5-mini	3.8B	50.2	44.6	37.0	41.4
RAG (BGE)	Llama-3.1	8B	48.3	46.7	45.7	47.2
	Llama-3.1	70B	53.3	50.0	48.4	48.8
	Gemma-2	9B	49.2	36.5	41.6	42.8
	Phi-3.5-mini	3.8B	46.2	43.9	36.0	44.8
RAG (Contriever)	Llama-3.1	8B	47.4	47.7	43.6	48.2
	Llama-3.1	70B	51.0	51.0	47.4	48.8
	Gemma-2	9B	48.5	40.8	38.8	41.8
	Phi-3.5-mini	3.8B	44.2	44.6	39.0	41.4
IterKey (BM25)	Llama-3.1	8B	62.0	49.7	50.2	53.3
	Llama-3.1	70B	63.5	54.9	53.8	57.7
	Gemma-2	9B	36.1	26.1	35.9	36.1
	Phi-3.5-mini	3.8B	47.9	42.7	36.8	41.2

Table 18: 'Vanilla' uses no retrieval. 'RAG (BM25)', 'RAG (E5)', 'RAG (BGE)', and 'RAG (Contriever)' apply a single retrieval step based on the original query, using BM25, E5, BGE, and Contriever respectively. Our proposed 'IterKey (BM25)' iteratively generates and refines keywords, performing up to five retrieval iterations to optimize answer quality.

Method	Model	Size (B)	Entity QA	HotpotQA	Natural QA	WebQA
Vanilla	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
RAG (BM25)	Llama-3.1	8B	54.0	47.0	44.8	51.4
	Llama-3.1	70B	54.6	46.2	43.4	47.4
	Gemma-2	9B	47.9	39.6	33.2	41.6
	Phi-3.5-mini	3.8B	48.2	42.2	32.6	40.2
ITRG Refine (E5)	Llama-3.1	8B	60.6	53.4	53.6	56.2
	Llama-3.1	70B	57.1	52.9	53.3	51.6
	Gemma-2	9B	54.2	47.6	47.4	48.5
	Phi-3.5-mini	3.8B	54.3	47.1	36.2	45.6
ITRG Refresh (E5)	Llama-3.1	8B	58.5	49.2	52.2	56.3
	Llama-3.1	70B	60.1	53.2	55.7	52.2
	Gemma-2	9B	50.2	40.8	45.9	49.3
	Phi-3.5-mini	3.8B	49.3	43.9	36.0	41.4
IterKey (BM25)	Llama-3.1	8B	62.0	49.7	50.2	53.3
	Llama-3.1	70B	63.5	54.9	53.8	57.7
	Gemma-2	9B	36.1	26.1	35.9	36.1
	Phi-3.5-mini	3.8B	47.9	42.7	36.8	41.2

Table 19: 'Vanilla' does not use retrieval. 'RAG (BM25)' apply a single retrieval step based on the original query, using BM25. 'ITRG Refine (E5)' is a method that utilizes E5 for iterative refinement. 'ITRG Refresh (E5)' also uses E5, performing refinement based on query refreshment. The proposed method, 'IterKey (BM25)', iteratively generates and refines keywords, performing up to five retrieval iterations to optimize answer quality.

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

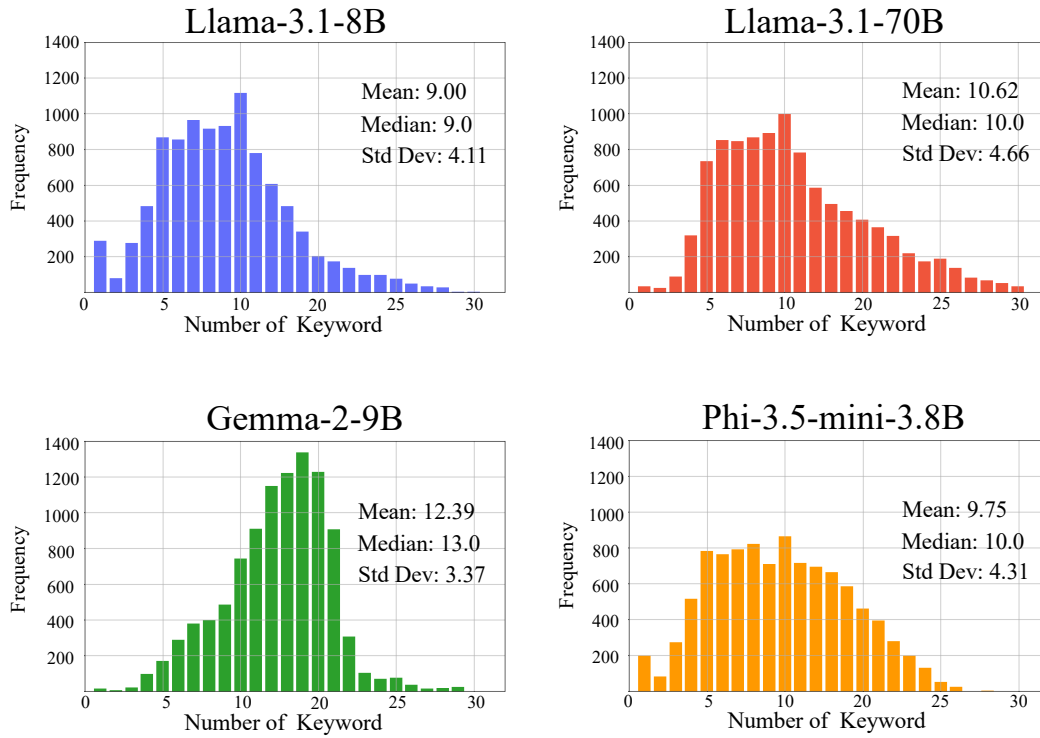Table 20: Results of different models across iterations, displaying both the mean and variance.

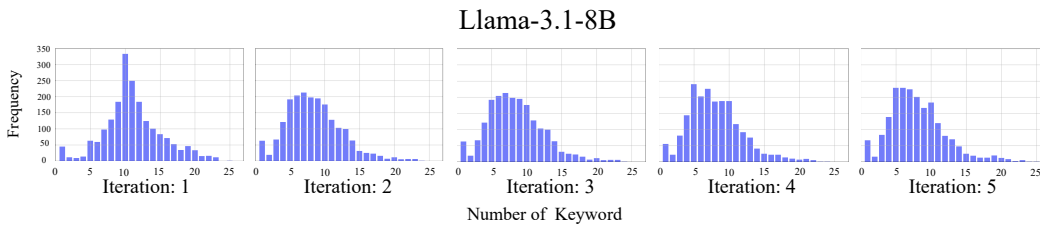Figure 3: The distribution of keywords generated by each LLMs.

Figure 4: The distribution of the number of keywords per iteration in LLama 3.1-8B.
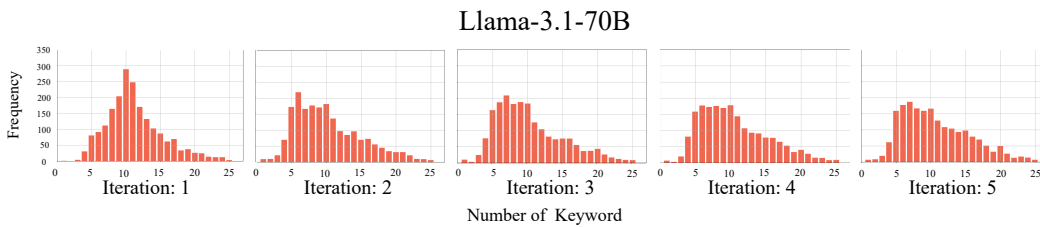

Figure 5: The distribution of the number of keywords per iteration in LLama 3.1-70B.
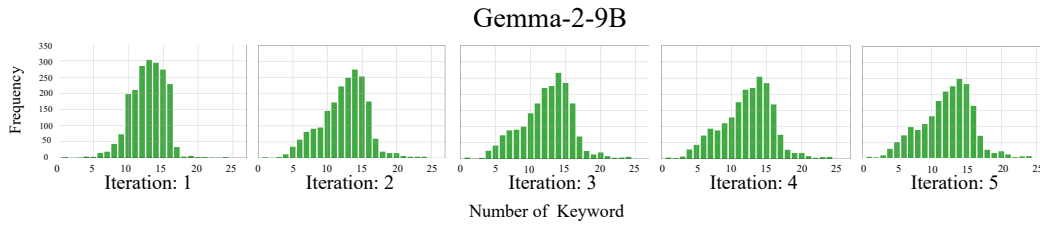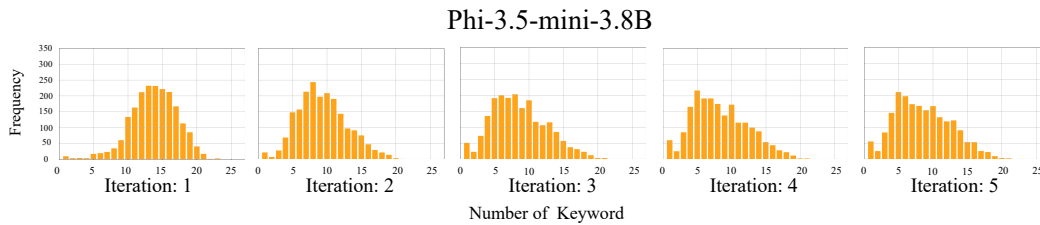
26

Gemma-2-9B

Figure 6: The distribution of the number of keywords per iteration in Gemma-2.

Phi-3.5-mini-3.8B

Figure 7: 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 21: 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, 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	Newark, 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 22: Case 1: 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 23: Case 2: Correct Answer with False Validation.

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 24: Case 3: Model fails to verify the correct answer despite arriving at it.